Keypoint-based scene-text detection and character classification using color and gradient features

Sriman, Bowornrat

DOI:
10.33612/diss.118694101

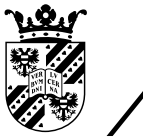
Keypoint-based scene-text detection and character classification using color and gradient features

Bowornrat Sriman

ISBN printed version: 978-94-034-2496-5
ISBN electronic version: 978-94-034-2495-8

Printed by: Gildeprint
Cover idea: Potchara Pruksasri

This research was supported by the Netherlands Fellowship Programmes (NFP)
under grant no. CF8777/2013,
University of Groningen, the Netherlands,
Mahasarakham University, Thailand and
College of Asian Scholars, Thailand.



university of groningen

Keypoint-based scene-text detection and character classification using color and gradient features

PhD thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus Prof. C. Wijmenga
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Friday 28 February 2020 at 16.15 hours

by

Bowornrat Sriman

born on 8 May 1978
in Khon Kaen, Thailand

Supervisor
Prof. L.R.B. Schomaker

Co-supervisor
Dr. C. Jareanpon

Assessment committee
Prof. G.A. Fink
Prof. D. Gavrila
Prof. R. Carloni

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Text localization and classification in natural scene images is receiving an increasing amount of attention in computer vision and artificial intelligence (AI). This is not surprising since the problem has many facets. For instance, one may wonder whether the mechanisms in human perception are useful in this application. How to implement a working system, using current perspectives on image processing? What are the disturbing factors we can expect in the area of image processing? Are methods generic or is there a dependency on script types, e.g., Western versus Asian? Finally, if such methods would exist, how can they be put to good use in real applications?

Although AI methods currently cannot compete with human eyes in terms of accuracy, AI also has strong points, such as a consistent response and scalability in terms of handling streams of visual data in self-driving cars, personal body cams in surveillance or offline processing of large numbers of scene images. However, the human vision system solves one essential bottleneck processing step very effectively. Instead of processing a complete image in high resolution, human vision uses selective attention. This is a behavioural and cognitive process of selectively concentrating on a distinct aspect of information in the visual array. The psychological concept of attention is described by William James (James 1890) as "*the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought...It implies withdrawal from some things in order to deal effectively with others*". Tsotsos et al. presented a model of visual attention based on the concept of selective tuning (Tsotsos et al. 1995), which is accomplished by a top-down scheme in which there are four basic components of the visual attention mechanism. The performance of the proposed model is highly appropriate for solving problems in a robotic vision system.

Visual attention, in summary, is the cognitive process of actively concentrating and directing the mind to an object which has an explicit structure or model, and disregarding anything irrelevant. This avoidance of unnecessary visual computing is desirable because in many applications, the computing and energy resources are limited, and the processing of large images becomes costly. If we want to implement a visual attention mechanism, a mathematical procedure is required to realize an attention-based model using the features of the image helpful in understanding the presented visual patterns.

In technical computer-vision systems, it is common to define a number of discrete processing steps. The processing pipeline starts with Image Acquisition, i.e., the process of sensing the image data using a visual sensor device. The next stage, Image Preprocessing, requires image size adjustment and noise reduction, that takes place before actually analyzing the image in the vision system. Visual text patterns that are presented in real-world scene comprise various colors, variations in text layout, patterns, multilingual content, and font styles, low resolution, and uneven illumination. The perspective distortion will geometrically affect large characters that are nearby, as well as small character patterns, that are presented at a higher distance from the camera. These problems are challenging for text localization and classification of natural scenes using mobile devices.

Many studies have been conducted to solve these problems. However, most of them are based on English script and not on complex Thai script, which consists of four vertical zone levels in the written line, and the meaning of a word is not correct if details are missing in any of the levels. The structure of Thai characters also contains complex geometric shapes including lines (vertical/horizontal/tilted), circles, and curves. Some letters contain line crossings and points of branching.

Additionally, the increased proficiencies of mobile phones support image processing capabilities on mobile devices. This enhances the opportunity for image acquisition and processing anywhere, anytime, which makes it easy to detect and classify foreground (text) and background (non-text) in different environments. Moreover, the development of computer vision and pattern recognition technology makes it more feasible to solve geometrical problems of image processing. If the detected text is recognized (decoded) properly, it can be employed in information retrieval, text to speech, text translation and other tools.

Of the aforementioned facets, we will now focus on the challenges of image processing in the next section.

1.1 Challenges of image processing

The human visual system is so effective that we do not notice difficulties and problems that the natural system solves for us. Digital cameras are quite similar to human eyes because they can capture images but they also have many functions such as macro capturing, landscape capturing and auto-white balancing. However, in technical systems, a number of correction and normalization functions need to take place to achieve similar robustness to the human visual system. Therefore, there are a variety of geometrical problems of image processing on both mobile devices and processor pipelines for text recognition, as shown in Figure 1.1. An analog image is captured by charged-coupled devices (CCD), which has over 1,000,000 pixels (1 Mega-pixel), in contrast to a pair of normal human eyes, which have approximately 240,000,000 pixels and can see approximately 10 million different colors. The most commonly used CCD color filter array type is RGGB or red-green-green-blue (the Bayer pattern). This pattern is used because the human eye is particularly sensitive to green (*CCD (Image Sensor) Design* 2019). The analog image is converted by an analog-to-digital (A/D) converter into digital image data. The image preprocessing performs a variety of calculations on a huge amount of digital image data through a color image that can be further used in the image processing process.

1.1.1 Geometrical problems of image processing

The study will start with processing on a mobile device by capturing 245 natural scene images of sign boards and banners from 31 places (in the early morning and afternoon), from 7 angles in each place (the examples are shown in Figure 1.2). Geometric problems such as perspective and lens distortion, various text sizes and colors, and reflection could be encountered.

- *Reflection*: When light strikes on an object, the impact area generally becomes brighter where the object reflects the light beam. This causes text on the surface of the object to be blurred and unclear. Even the human eye will have difficulty

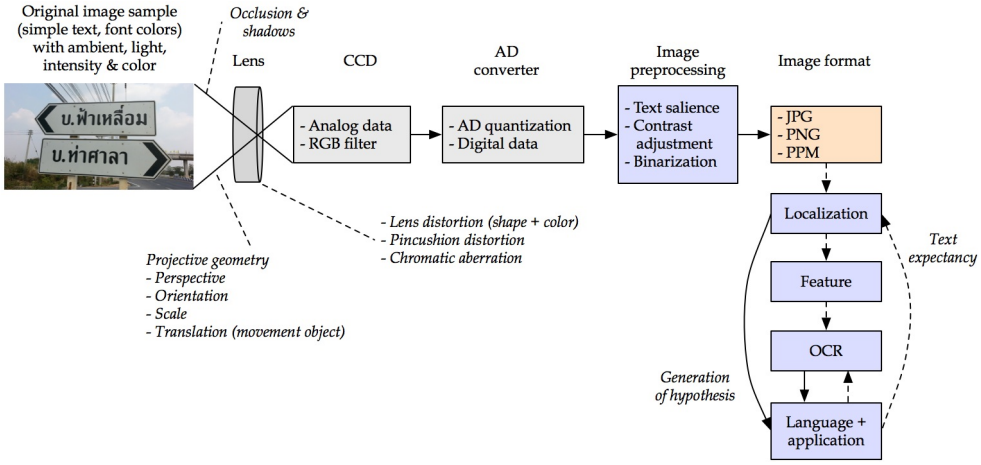


Figure 1.1: Geometrical problems of image processing on mobile devices and processor pipeline for text recognition. The purple boxes represent the major forms of the processing efforts in this study.

(a) Reflection. (b) Occlusion & shadows. (c) Perspective distortion.

(d) Scale. (e) Character segmentation. (f) Low resolution.

(g) Nonlinear warping. (h) Complex background.

Figure 1.2: Sample pictures of geometrical problems in image processing on mobile devices for text localization, classification, and recognition.

reading it. This problem is more challenging when a mobile device is used to take a picture of an object in this scenario. The process of segmenting the character from the picture becomes a major challenge (Figure 1.2a).

- *Occlusion & shadows*: Likewise, when light strikes on an object, it will be deflected and turned into shadow. This might be the shadow of a tree, pole, shelves etc., and it will occlude the object and cause an incomplete message. Therefore, parts of the same letter that are in different inside and outside lighting conditions will have different intensities and color tones and cannot be identified (Figure 1.2b).

- *Perspective distortion*: Pictures of a natural scene are usually actual images which are caused by the direction of the shooting: the angle of depression, elevation, left side and right side. The effect is that the image border appears to be an isosceles trapezoid. This image was taken at 15 degrees from the sign which is deep and narrow. The higher the degree, the deeper and narrower the sign will look. At the same time, characters on the sign will also become thin and short (Figure 1.2c).

- *Scale*: This is the ratio of the image length, e.g., scale1:1,000 = one unit long instead of the actual length of 1,000 units. Suppose that an image with 1,200×800 pixels is displayed at 100% on a screen display with 600×400 pixels, then the image will overfill a screen whose scale is 1:1. The scale will be minimized two-fold to 1:2 to fit the display scale. On the other hand, if an image with 300×200 pixels is extended to fit the full screen, it will have a ratio of 1:0.5 and this will cause the image to be distorted. Consequently, whether the image is zoomed in or out, the resolution of the image is lost (Figure 1.2d)

- *Character segmentation problems*: Even though Thai characters do not have lowercase and uppercase as in English, Thai words are divided into 4 vertical zone levels in a written line. In many cases, characters that appear on the packaging of various products are formed of diverse letters; for example, they are freestyle or contain 3D letters etc. Some product names are all the same font size or they alternate between larger and smaller on one line. It is therefore very difficult to segment the letters (Figure 1.2e).

- *Low resolution*: Pictures taken by mobile devices often have low resolution because of the small number of pixels which causes the picture to be blurred. On the other hand, typical digital cameras have high resolution due to a large number of pixels, and thus produce sharp images. Therefore, the resolution of the image is affected by the image processing (Figure 1.2f).

- *Nonlinear warping*: The properties of the camera lens or the objects of interest cause distortion of the image. For instant, letters on a bag that are warped by the shape of the bag cause incomplete characters and missing letters. As a result the performance of text recognition will be reduced (Figure 1.2g).

- *Complex background*: Photographs of a real place usually consist of trees, windows, electric poles, shadows, reflections of the sun by shiny objects, etc. If it is difficult to extract characters from a simple document, then an uncontrolled environment is even more difficult. It is challenging to find a text area and separate the foreground from the background in a normal scene picture (Figure 1.2h).

1.2 Background of image processing on mobile devices

1.2.1 Comparison of digital cameras and mobile devices

There is fierce competition in the mobile device market in terms of low pricing as well as software and hardware development in order to support the needs of the consumer. One of these developments is photography, referred to in an article by Caroline Cakebread (Cakebread 2017) which stated that the number of digital photos taken on smartphones in 2017 was 1,200 billion pictures, as illustrated in Figure 1.3. Furthermore, sales of digital cameras have drastically declined over the years and been replaced by smartphones that were used eight times as often as digital cameras in 2017. This is because photos or files taken by a smartphone are much easier to upload to social media sites including Facebook and Instagram, and therefore stand-alone cameras are no longer popular. Mobile devices today are almost equivalent to a personal computer. Their power efficiency reduces power consumption and increases usage time which satisfies consumers. However, Table 1.1 shows

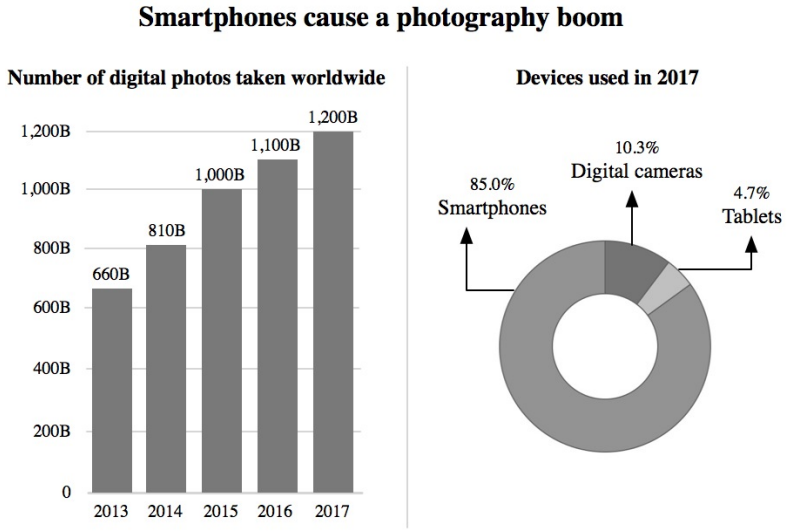


Figure 1.3: Smartphones are causing a photography boom that is turning millions of people around the globe into prolific photographers (left). The comparison of devices used in 2017 shows their usage in descending order: smartphones (85.0%), digital cameras (10.3%), and tablets (4.7%) (right) [redrawn after] (Cakebread 2017).

that mobile devices are still limited when compared to general digital cameras because of their low resolution, a smaller sensor, more salt-and-pepper noise, motion errors and a lack of focus. Also, the image processing on a mobile device still requires a high-speed CPU to produce a real-time result.

1.2.2 Trends of image processing on mobile devices

The mobile device is becoming increasingly popular for natural data sensing because it has plenty of storage capability and portable digital imaging devices. It is also versatile, e.g., video recording and image capturing. In addition, there are various image applications (apps) on mobile devices that are very helpful, including text to speech conversion for visually impaired persons, real-time text translation support for foreign tourists, handwritten music notation recognition, and road-sign detection and tracking for driver assistance. Some of the applications combine a mobile phone, photos and social networks together, capturing and sharing images

Table 1.1: The difference features of digital cameras and mobile devices.

Features	Digital cameras	Mobile devices	Features	Digital cameras	Mobile devices
Sensor	CMOS,BSI-CMOS and CCD. Size between 4.54×3.42 mm. 36×24mm.	CCD and CMOS	Resolution	Wide range 640× 480-6,016×4,000 pixels. Higher resolution and larger sensor camera.	Higher resolution but smaller sensor camera.
Zoom	Flexible adjustment: optical zoom 3x-35x and digital zoom 2x - 10x	Limited	Focus	Focus range 30 - 80 cm. and aperture range f1.4 - f16, auto focus and manual focus.	Auto focus and fixed focus
Shooting performance	Better than mobile devices, easy to use and adjustable shooting.	Fast motion capture is still a problem of mobile phone devices.	Salt-and-Pepper noise	Can be controlled manually.	Too much, especially in insufficient light conditions.
Blur	Usually sharp images.	Motion and out-of-focus errors are very common.	Video	Several modes support shooting in slow motion.	Spatial and temporal resolution of the camera is limited.
Portability	Bad	Good	Usability	Low, cannot continually process images.	High, continuous processing of image.

to a social network (such as Instagram). Bar and QR code product reader apps are also becoming available, and they have the flexibility to read at different distances from the bar and QR code. (Figure 1.4).

(a) QR code

(b) Face recognition

(c) Road sign recognition

Figure 1.4: Examples of image applications on mobile devices.

There are examples of studies on processing on mobile devices such as PocketPal and PocketReader applications, which are implemented on a generic framework called OCR-droid (Joshi et al. 2009) and digitize text in real time using mobile phones. Their experimental results of binarization achieved 96.94% in English script accuracy under normal lighting conditions, but show that its performance is degraded in poor or flooded lighting. In the case of skewed text detection and

auto-rotation it works reasonably well with an image rotation of up to 30 rotations in both clockwise and counter-clockwise directions, but the performance drops sharply with an image rotation of 35. It takes a maximum of 11 seconds to complete the whole process.

A restaurant recognition system (Herranz et al. 2016, Song et al. 2014) using image processing and a social network runs on the iPhone; it captures an interesting restaurant then processes and analyzes which restaurant the photo is taken from. The user receives real-time information on the restaurant, e.g., menu, price, comments from other users, or promotions from the internet. Furthermore, users are allowed to leave and share a comment on the social network. This process comprises the blur and finding features of an image by using Gaussian and scale invariant feature transform, respectively. The experimental result of 35 images from 7 English restaurants is that 74.28% of them can be classified correctly.

Text recognition on mobiles for the visually-impaired person (Dumitras et al. 2006) can be used on a mobile phone based on client-server architecture. The server sends the extracted English script back to a Nokia 6620 (testing equipment), which is then displayed and pronounced using a speech-synthesis engine to the user. They demonstrate with multiple images of an outdoor sign taken from various distances and angles. The level of recognition reduces if the character size is less than approximately 21 pixels, or the text angle is 30% or greater.

A camera-based equation solver for Android devices (Sikka and Wu 2012) is efficient for solving simple computer-printed expressions and handwritten expressions such as addition, subtraction and multiplication; the user captures a camera image of the equation and the camera displays the solution. Text recognition is divided into two parts: computer-printed expressions are processed with Tesseract and handwritten expressions are processed with a support vector machine (SVM). The experimental results of recognition by Tesseract and SVM are both approximately 85-90%.

Real-time translation applications on mobile devices are being continuously developed; most of the texts are recognized by Tesseract and automatically translated

by Google Translate. For example, English translated into Chinese on Android [1]; here the text feature filtering and the text region binarization are a combination of connected components and the Otsus method, and the performance of text extraction, recognition and translation has a correct character-recognition rate of higher than 85%. English to Spanish translation has been proposed by Canedo-Rodriguez et al. (Canedo-Rodriguez et al. 2009). Their system shows a high robustness for detection, binarization, recognition and translation under uneven illumination situations. It can handle foreground-background color changes due to lighting reflections and even reverse foreground-background images. The authors claim that the application works in a very short amount of time, which ensures its viability.

A real-time translation system (Fragoso et al. 2011) on a Nokia N900 smartphone camera requires the user to tap a particular word on the touchscreen in order to produce a translation. After that the system first detects the bounding box around the word, then the exact location and orientation of the text within. The performance of optical character recognition (OCR), translation, detecting text and color is relatively low. However, a fast automatic text detection algorithm has improved translation for a mobile augmented reality (AR) translation system on a mobile phone (Petter et al. 2011). This algorithm works on a gray scale image for faster processing time. The processes include three steps: finding a zone of interest that may contain a letter, verification of the zone of interest, and finding the rest of the word. The performance of this algorithm provides text detection accuracy with precision at 68% and recall at 87%. Another interactive text recognition and translation from Chinese into English on mobile devices (Hsueh 2011) using a translation web service runs on the Nokia N900 smartphone. The recognition performance is compared between Simplified Chinese text and English text, and the corrected auto-classification of phrase-matching accuracy and character-wise match accuracy for English are 44.0% and 34.0%, and for Chinese are 78.5% and 80.6%, respectively. However, Chinese OCR requires four times the time taken for a comparable English text on the same platform.

[1] https://stacks.stanford.edu/file/druid:my512gb2187/Ma_Lin_Zhang_Mobile_text_recognition_and_translation.pdf

1.3 Research questions

Q1: How to indicate where text lines are in a natural scene image?

Text is nearly omnipresent and related to all human life, for example, there is text on road signs, maps, and television. A scene image comprises several obstructing objects, for instance, luminance, a complex background and so on. The purpose of text line detection is to allocate a group of text components into candidate text regions with a small amount of background. Chapter 5, therefore, proposes modeling general text chunks and non-text using the SIFT feature. Further, a codebook-based classifier is presented that was created by an object histogram. The proposed technique has been evaluated using the text-detection processes of a Thai scene image dataset (TSIB). The results show that the proposed technique performs effectively in text localization in scene images, particularly for continuous-word scripts such as Thai but it is also beneficial for locating English scene text.

Q2: How can text and non-text blocks be identified?

This question is addressed in Chapter 3 and Chapter 4. Chapter 3 considers the essential features (object description, color distribution, and gradient strength) of an image that indicate the text and non-text properties. Using these features, the results of classifying text and non-text showed some successful results but also found that the color attribute provides an inadequate classification result due to illumination and other problems. Therefore, Chapter 4 proposes a novel technique to handle classification using the advantage of autocorrelation in a form of time series analysis. This describes the correlation between data sequences in a single series of sample values, at several delays in time. There are three kinds of time series: acf, ifft, and xcorr used for describing color information in an intensity-invariant manner for creating a histogram of nine different color intensities. This method can improve classification accuracy from the methods presented in Chapter 3. Using the object attribute that was extracted by SIFT gives a better result than the other properties in Chapter 3. Therefore CAH is combined with the SIFT feature to increase the efficiency of text and non-text classification. It appears that the combined features achieve a better result.

Q3: Is SIFT usable for both text detection and character recognition?

Natural urban scene images contain many problems for character recognition such as luminance noise, varying 2D, and 3D font styles. In addition, a character recognition problem in Thai script is that there are characters that are similar in consonants and vowels. Therefore, Chapter 2 presents a model of textuality of a region of interest based on object attention patches using the scale invariant feature transform (SIFT) algorithm. The point of interest (POI) by SIFT in each character is substantial because it is an indicative feature of a character. This model can solve the problem of similar-shaped characters. In the preliminary test, the character models can also detect characters in scene-text images using mesoscale attentional patches. Therefore, the proposed model is usable for scene-text detection and recognition purposes.

1.4 A survey of recent research in the field

1.4.1 Light, low-contrast and luminance noise

The contrast and luminance noise are uncontrolled factors in natural scene images, unlike a scanned document input which has similar quality. The Sauvolas local binarization algorithm combined with the background surface thresholding algorithm to handle uneven lighting conditions were applied by Joshi et al. (Joshi et al. 2009). The algorithm worked better and faster than the Otsu algorithm or the Niblack algorithm. In order to avoid blur or lack of focus, the study by Joshi et al (Joshi et al. 2009). adopted the AutoFocus API provided by Android SDK, it can handle focusing on the text sources automatically.

A morphological reconstruction technique (Trémeau et al. 2011) can be applied to remove dark/light objects connected to borders which are darker/lighter than their surroundings. In order to suppress lighter objects the zero is marked on every pixel of the image except the border. These experiments decrease the background intensity variations and enhance the text layers of the image. Otherwise its borders are set to zero before applying the connected closing transformation which is used to suppress darker objects, but the result of this is ineffective.

Symmetrical and asymmetrical fuzzy algorithms are used to reduce noise in im-

age sequences (Saeidi et al. 2008). The output of the algorithms is calculated from the intensity values of noise pixels and the corresponding weights. They performed in low, medium and high-density salt-and-pepper noise in the picture. The noise was removed very efficiently. The image structures and its edges were preserved by the fuzzy algorithm(s), which are more effective than the concatenated median (CM) or the center weighted median (CWM) filters.

There are many papers to solve impulsive noise problems, and a new image-filtering technique (Smolka et al. 2002) solves the problem of impulsive noise in color images using the concept of maximizing the similarities between the pixels in a predefined filtering window. The filter efficiency is estimated by three quantitative measures: the root-mean-square error (RMSE), the normalized mean square error (NMSE) and the peak signal-to-noise ratio (PSNR), which show very good results when compared with standard techniques. However, the computational complexity of the vector median filter (VMF), the basic vector directional filter (BVDF) and proposed filter has the same rank as O(n^4). A coplanar filter through which impulsive noise can be completely removed, and small variations are eliminated causing piecewise smoothing and maintaining sharp edges, was proposed by Fan et al. (Fan et al. 2001). An experimental comparison between no-filter, Gaussian, median and the Coplanar filter shows that the Coplanar filter performs better than the other methods.

1.4.2 Complex backgrounds

Natural scene images include a variety of color which can cause difficulty in identifying an object in a picture using a computer. Color-based components have been used to extract the foreground from the background (Park et al. 2007) by ordinary methods for segmenting text image based on chromatic and achromatic components. They analyzed and compared the performance of five color components R, G, B, H, S and I in scene images. Their experiment with various RGB images performed the segmentation based on the proposed method and showed that it is robust in the case of highlighted natural scene images even though there are many separate regions in the characters. Connected components in conjunction with color quantization are used to extract text regions from natural scene images to reduce bit color images (Li et al. 2001). The experimental results yielded a detection rate of

84.55% and a false-alarm rate of 5.61%.

Solving the problem of text retrieval from complex backgrounds by using a touch-screen interface has been addressed in benchmark datasets: ICDAR2003 and the KAIST scene text database (Jung et al. 2011). Nevertheless, there have been problems with substantial reflection effects, small text regions because of inadequate resolution, a variation in stroke-width of character, minor differences between the text and background, and considerable color change within a single image component. However, the authors insist that the problem of extracting characters from a complex background can be solved. Chiung-Yao Fang et al. (Chiung-Yao et al. 2003) illustrate a method for detecting and tracking road signs from video images with complex backgrounds to extract color and shape features using two neural networks, respectively. They used a process characterized by fuzzy-set discipline to extract possible road signs, and used a Kalman filter for the tracking phase to predict the positions and sizes. The performance of the proposed method is both accurate and robust.

1.4.3 Variant text patterns

Texts in common scene images follow various patterns. In order to identify text from a complex background, first it is necessary to know the text position. The Canny edge detector is used to compute edges in the image and then pixels are grouped into letter candidates using stroke width transform (Epshtein et al. 2010). The performance of the proposed text detection algorithm tested on the ICDAR 2003 and ICDAR 2005 is 79.04% for the word recall rate, 79.59% for the stroke precision, and 90.39% for the pixel precision ratio. The new image feature thus obtained can then be applied to many languages and fonts. Stroke Gabor Words uses Gabor filters to describe and analyze the stroke components in text characters or strings to detect text in natural scene images (Yi and Tian 2011). The proposed algorithm is evaluated on two datasets: the first dataset is the ICDAR 2003 and the second dataset is provided by Epshtein et al. (Epshtein et al. 2010) The performance of the algorithm from the detected text regions is compared with the ground truth text regions. The algorithm results are 0.64 for precision, 0.76 for recall and 0.68 for the f-measure, which demdemonstrate that it can handle complex backgrounds and variant text patterns.

1.5 Organization of this thesis

This research will contribute to the development and analysis of methods and demonstration of their performance for character recognition, text/non-text classification, and text localization. The research will be conducted in three main parts. Chapter 2 introduces the paradigm of the bag-of-visual words. Chapter 3 and Chapter 4 cover the problem of text and non-text classification. Chapter 5 demonstrates text localization in scene images using a codebook-based classifier.

Chapter 2 introduces the paradigm of the bag-of-visual words using the SIFT feature to create character models that are usable for expectancy-driven techniques. The produced models (object attention patches) are evaluated regarding their individual provisory character recognition and used in preliminary experiments on text detection in scene images. The results tested on ICDAR2003 plus Chars74K, TSIB, and Thai NECTEC datasets show that the proposed model-based approach can be applied to a coherent SIFT-based text detection and recognition process.

Chapter 3 presents an alternative classification method based on three categories of object-attribute features, namely the object description, color distribution and gradient strength. Each feature is computed into a classifier model. The robustness of this method has been tested on the ICDAR2015 dataset. The experimental results show that the performance of the proposed method performs competitively against other state-of-the-art methods.

Chapter 4 proposes a novel adjoined feature between the color autocorrelation histogram (CAH) and scale-invariant feature transform (SIFT) for scene text classification. Parameter tuning is performed and evaluated, in which the best result will be chosen to create the CAH classifier and SIFT classifier. There are nine color spaces using the selection criteria for creating the CAH feature, which are compared to get the best color classification result. As regards the final classifying methods, the regular nearest neighbor (1NN) and the support vector machine (SVM) were compared. The performances of the proposed model appear to be robust and suitable for Asian scripts such as Kannada and Thai, where the scene text fonts are characterized by a high curvature and salient color variations.

Chapter 5 shows text localization in scene images using a codebook-based classifier. The proposed method has been evaluated for the text-detection and classifying

processes using the TSIB dataset. The results show that the proposed technique improves the text localization particularly for a continuous-word script (Thai) which is rich in ornaments. The proposed technique is also beneficial for locating the text of the ICDAR2003 dataset (English script).

Chapter 6 provides a discussion and concludes this thesis, answers the research questions, and suggests some further research directions.

Part I

Points of attention for text detection

This chapter is an adaptation of paper:

Sriman, B. and Schomaker, L. – *"Object attention patches for text detection and recognition in scene images using sift,"* ICPRAM 2015 - 4th International Conference on Pattern Recognition Applications and Methods, Proceedings, Vol. 1, pp. 304-311, 2015.

Chapter 2

The paradigm of the Bag of Visual Words

Abstract

This chapter presents a modeling approach that is usable for expectancy-driven techniques based on the well-known SIFT algorithm. Due to several techniques have been proposed based on a bottom-up scheme, which provides a lot of false positives, false negatives, and intensive computation. Therefore, an alternative, efficient, character-based expectancy-driven method is needed. The produced models (Object Attention Patches) are evaluated regarding their individual provisory character recognition performance. Subsequently, the trained patch models are used in preliminary experiments on text detection in scene images. The results show that our proposed model-based approach can be applied for a coherent SIFT-based text detection and recognition process.

2.1 Introduction

Optical Character Recognition (OCR) is an important application of computer vision and is widely applied for a variety of alternative purposes such as the recognition of street signs or buildings in natural scenes. To recognize a text from photographs, the characters first need to be identified, but the scene images contain many obstacles that affect the character identification performance. Visual recognition problems, such as luminance noise, varying 2D and 3D font styles or a cluttered background, cause difficulties in the OCR process as shown in Figure 2.1. In contrast, scanned documents usually include flat, machine-printed characters, which are in ordinary font styles, have stable lighting, and are clearly against a plain background. For these reasons, the OCR of photographic scene images is still a challenge.

Figure 2.1: Sample pictures of visual recognition problems in scene images for text recognition.

Many techniques to eliminate the mentioned OCR obstacles in scene images have been studied. For example, detecting of and extracting objects from a variety of background colors might be partly solved by color-based component analysis (Li et al. 2001, Park et al. 2007). The difficulty of text detection in a complex background can be overcome by using the Stroke-Width Transform (Epshtein et al. 2010) and Stroke Gabor Words (Yi and Tian 2011) techniques. In addition, contrast and luminance noise are uncontrollable factors in natural images. Several studies (Fan et al. 2001, Smolka et al. 2002, Zhang et al. 2009) have been conducted to conquer these problems regarding light.

However, the aforementioned methods act bottom-up and are normally based on salience (edges) or the stroke-width of the objects. In a series of pilot experiments, we found that the results present a lot of false positives or non-specific detection of text (Figure 2.2), and the recall rates are also not very good. Hence, a more powerful method is needed.

Looking at a human vision, expectancy plays a central role in detecting objects in a visual scene (Chen et al. 2004, Koo and Kim 2013). For example, a person looking for coins on the street will make use of a different expectancy model than when looking for text in street signs. An intelligent vision system requires internal models for the object to be detected (where the object is) and for the class of objects to be recognized (what the object is).

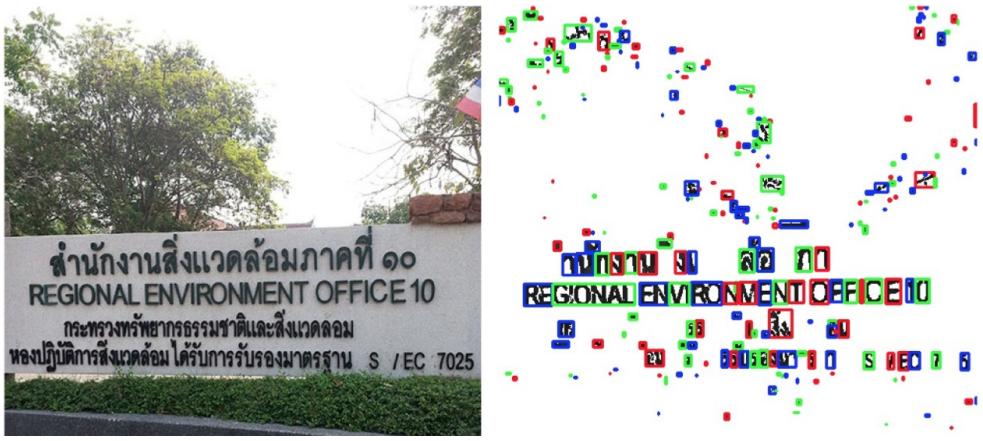


Figure 2.2: Example of the Stroke-Width Transform result on Thai and English scripts.

A simple modeling approach would consist of a full convolution of character model shapes along an image. Such an approach is prohibitive: it would require to scan for all the characters in an alphabet, using a number of template sizes and orientation variants. All of these processes would make the computation too expensive. Therefore, a fast invariant text detector would be attractive. It should be expectancy- driven, using a model of text, i.e., the degree of 'textuality' of a region of interest. A well-known technique for detecting an object in a scene is the Scale Invariant Feature Transform (SIFT) (Lowe 2004). It is computationally acceptable, invariant and more advanced than a simple text-salience heuristic. Therefore, we address the question of whether SIFT is usable for both text detection and character recognition.

2.2 Model for object attention patches using SIFT feature

Text detection based on salience heuristics often focuses on the intensity, color, and contrast of objects appearing in an image. The salient pixels are detected and extracted from the image background as a set of candidate regions. Saliency detection

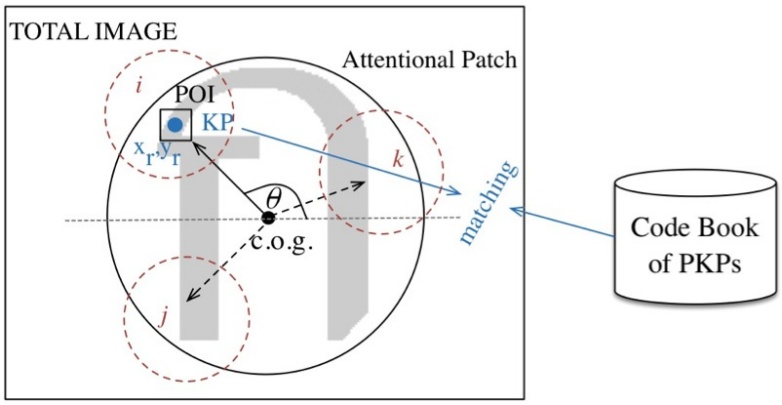


Figure 2.3: Schematic description of the attentional-patch modeling approach. Center of gravity (c.o.g.) $\triangleq$ (0,0)

is a coarse textuality estimator at a micro scale, yielding the probability for each pixel that it belongs to the salient object (Borji et al. 2015), while the information such as luminance and color space (e.g., RGB) is of limited dimensionality. In the text-detection process, the set of candidate regions is merged with its neighbors and then processed in a voting algorithm to eliminate non-text regions before presenting the final outputs of the text regions. Even then, there may still exist a lot of false positives and false negatives (cf. Figure 2.2).

We propose to increase the information used for the 'textuality' decision by using a larger region, at the mesoscale, i.e., the size of characters. In this way, the expectancy of a character is modeled by attentional patches. The type of character modeling proposed here serves two purposes: detection and recognition. The requirements are that the process should be reasonably fast and able to handle variable sizes and fonts. This can be realized by exploiting the detection of small structural features, such as is done in SIFT-like methods, in combination with modeling the expected 2D layout of these key points in characters.

For each character in an alphabet are computed SIFT keypoints. The keypoints are usually highly variable. In order to reduce the amount of modeling information, clustering is performed on the 128-dimensional key points (KP) SIFT descriptors, per character, yielding a codebook of prototypical keypoints (PKP). The center of gravity (c.o.g., x, y) over all the keypoints for a character is computed, as well as the

densities for PKPs in the character patch. This yields the expected relative (x_r, y_r) position of a PKP for this character, dubbed the point of interest (POI). The spatial relation of the PKP positions allows the expected PKPs j at relative positions and angles to be modeled, given a detected PKP i and an expected character c. Figure 2.3 provides a graphical description of the model. The evidence-collection process starts with the keypoint extraction, entering a scoring process for both 'textuality' and the likelihood of a character present at the same time.

2.2.1 Scale invariant feature transform (SIFT)

The SIFT technique is developed to solve the problem of detecting images that are different in scale, rotation, viewpoint, and illumination. The principle of SIFT is that the image will be transformed to scale-invariant coordinates relative to local features (Lowe 2004). In order to obtain the SIFT features, it starts with finding scale spaces of the original image, the Difference-of-Gaussian (Burt and Adelson 1983) function is computed to find interesting keypoints, scales, and orientation invariances. Next specifying the location where the exact keypoint is, an interesting point will be compared to its neighbors that then roughly presents maxima and minima pixels in the image. These pixels can be used to generate sub-pixel values to improve the quality of the keypoint localization using the Taylor expansion algorithm. The improved keypoints are better in matching and stability due to this technique. However, some low-contrast keypoints located along the edge, which are considered to be poor features will be eliminated.

After receiving the keypoints, the local orientation of each keypoint will be assigned by collecting gradient directions and then computing magnitude and orientation of the pixels around that keypoint. The result will be put into an orientation histogram, which has 360 degrees of orientation and then divided into 36 bins. Any bin percentage that is higher than 80% (Lowe 2004) will be assigned to the keypoint. At the end, image descriptors are created. The descriptors are computed using the gradient magnitude and orientation around the keypoint. This calculation is executed from 16×16 pixels and grouped into 4×4 cells. Each cell will be used to form the 8-bin histogram. Finally, histogram values for all the cells will be combined into 128 descriptors and assigned as the keypoint descriptor.

An important parameter in SIFT is the distance ratio threshold. In the original

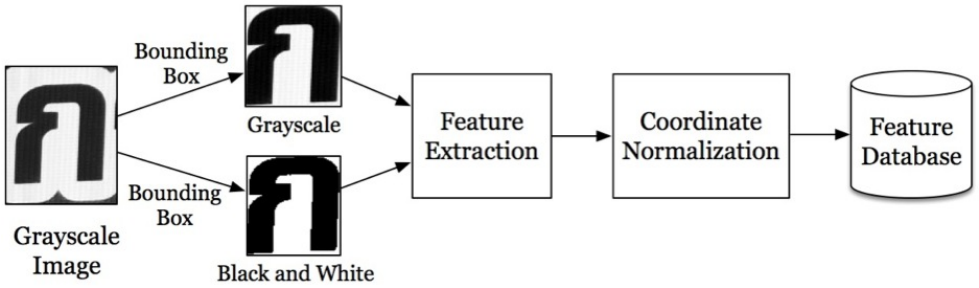


Figure 2.4: Process flow of feature extraction and coordinate normalization.

paper (Lowe 2004), an optimal value of 0.8 is proposed. However, the optimality of this threshold depends on the application. In training mode, false positive keypoints are the problem whereas in 'classification testing' mode, false negatives may be undesirable. Therefore, we will use different values for this parameter in the different processing stages.

2.2.2 Feature extraction and normalization

The character images are converted to grayscale in order to increase the speed and simplify the recognition process. Some character images are inverted if necessary to always have dark ink (foreground) and a light background. All the character images in each class are randomized into two sets: training and testing sets. Both of them will be processed in the feature extraction and the coordinate normalization methods. The extracted features will be used as the constituents of character models in the next step. The flow of this process is demonstrated in Figure 2.4.

Every grayscale image in the dataset is calculated for its bounding box of the character and is then cropped based on its detected box. The image is extracted features by SIFT that called keypoints (KP), which consist of the coordinate (x, y), scale, orientation, and 128 keypoint descriptors then collected into a database. In order to enlarge the number of KPs, they are also extracted from a binarized (B/W) copy of the character image. After receiving all the keypoints, the original source images are no longer needed in the process. Only the keypoint vectors will be utilized.

Since the absolute position of the character in the scene images is unknown, the local keypoints' positions needs to be in a relative scale. By the equations: $x' = \frac{x}{w}$

and $y' = \frac{y}{h}$ where w and h are character width and height, respectively. After that, the relative positions of the keypoint will be normalized to present in the same scale space as others by the equations: $x_{norm} = x' - 0.5$ and $y_{norm} = y' - 0.5$. Finally, the final keypoint vector consists of x_{norm}, y_{norm}, scale, orientation, and 128 keypoint descriptors.

2.2.3 Building prototypical keypoints (PKPs) using k-means

K-means clustering (Forgy 1965) is a well-known and useful technique to partition a huge dataset into a number of k groups, i.e., clusters. The members within the cluster have similar characteristics, and the average vector known as the centroid of the cluster is a good representative of the cluster. The centroid is expected to be the Prototypical Keypoint (PKP) of each keypoint cluster.

All the keypoints of each class are clustered into several groups using the k-means algorithm in the descriptor (N_{dim} = 128). We perform using values k = 300, 500, 800, 1,000, 1,500, 2,000 and 3,000 to produce various sensitivity levels of the model and then select the centroid of each cluster to be the descriptor of the PKP. We expect that in the 2D spatial layout, a distribution of the PKP's coordinates represents an important characteristic for each character class. However, defining the coordinate cannot make use of the average values of x and y since the clustering is performed in the descriptor of the keypoint. So, to determine the proper PKP's coordinate there needs to be a separate process.

Looking at a cluster of the keypoints from the previous step, the descriptor values of the keypoints are similar, but it is possible that the keypoints are located in different areas because the SIFT mechanism considers the prominent spots of an object in the picture. Some different parts of the same character may provide similar descriptor values. Therefore to find an appropriate x, y of the delegate PKP, we then perform the k-means clustering in the coordinate (x, y) within each cluster. Because of the small number of keypoints in the cluster, we use values k = 2, 3 and 4 to find the major area of the keypoints within the cluster. With k = 3, most results present an obvious major group of the cluster with a lower distribution rate than other k values. Therefore, we choose k = 3, and the centroid of the major cluster is selected as the PKP's coordinate (x_{pkp},y_{pkp}).

After that, the coordinate and the descriptor are combined to be a PKP of the

model. Algorithm 2.1 summarizes the steps to build a PKP. By, input: set of raw keypoints of characters, S_n = kp_{n1},kp_{n2},.., kp_{nm}. output: set of PKP of characters, F_n = pkp_{n1}, pkp_{n2},.., pkp_{nk}. Where m = raw keypoints in a class (1 to m); n = classes (1 to n); G = cluster of keypoints in descriptor; L = cluster of keypoints in coordinate.

Algorithm 2.1 Building PKP

```
for S_1 to S_n do
    G_{1..k} ← classify { S_desc,k-means,kgroups }
    for G_1 to G_k do
        pkp_desc ← getCentroid { G_desc }
        L_{1..3} ← classify { G_loc,k-means,3groups }
        L_max ← selectMajorGroup { L }
        pkp_loc ← getCentroid { L_max }
        pkp = { pkp_loc,pkp_desc }
    end for
    F = { pkp_1,pkp_2,..,pkp_k }
end for
```

2.2.4 Assigning a models point of interest

Given a general codebook with Prototypical Keypoints, it becomes essential to associate PKPs and their relative position to character models. The assumption is that each character has points of interest (POI) that elicit keypoint detection. The POI in each character is substantial because it is an indicative feature of a character. Therefore, we need to identify the interesting points of a model.

Based on the model generated in the previous section, we scatter its keypoints on spatial layouts to find the distribution of the model's features. The scatter diagrams (heat maps) in Figure 2.5 show that the normalized PKPs (bottom row) remain almost the same vital points (high density) of the character as raw KPs (top row in Figure 2.5). We assume that the heated area will represent spots of interest which are normally less than 10 per class according to our experimental results.

The PKP's locations (x_{pkp},y_{pkp}) are then clustered by doing the scan for k = 5, 6, 7, 8 and 9. We found that k = 7 provides the best centroids (x_{poi},y_{poi}), which are located in the proper area and can be considered as the POI. To complete the creation, every PKP of each cluster will be associated with the computed POI of the cluster. The POI assigning process is summarized in Algorithm 2.2. By, input: set of models features,

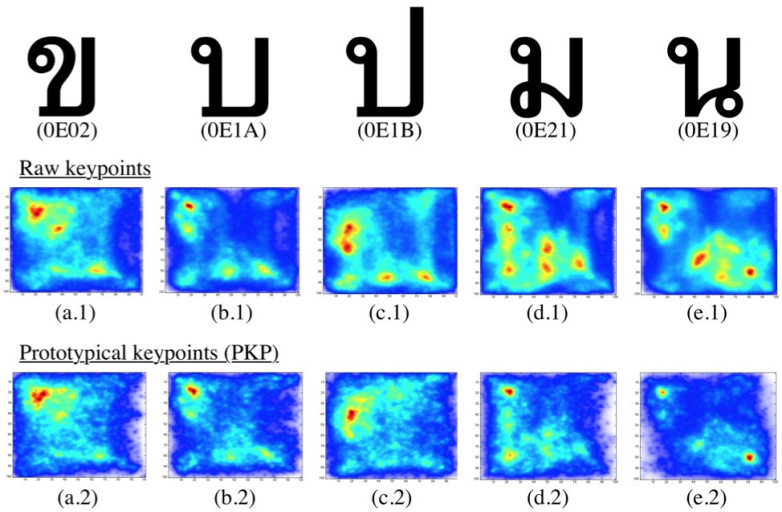


Figure 2.5: Samples of normalized PKP distribution of similar characters (with $K << N$ important keypoints are still retained).

Algorithm 2.2 Assigning POI to PKP

1: **for** F_1 to F_n **do**
2: $R_{1..t} \leftarrow$ classify { F_{loc},k-means,tgroups }
3: **for** R_1 to R_t **do**
4: $p \leftarrow$ getCentroid { R }
end for
5: P = { p_1,p_2,..,p_t }
6: **for** pkp_1 to pkp_k **do**
7: $p \leftarrow$ getMemberOf { pkp_{loc},P }
8: pkp = { pkp,p }
end for
9: F = { pkp_1,pkp_2,..,pkp_k }
end for

F_n = pkp_{n1}, pkp_{n2},.., pkp_{nk}. Where n = classes(1 to n); t = point of interest(1 to t); R = cluster of PKP ; P = set of POI. After computing this step, we will get a model structure elements comprising the Prototypical Keypoint (PKP) and their POIs as illustrated in Figure 2.6. The PKP coordinates are represented by the orange at its normalized location. The POI is marked by the yellow dot. The models are called *Object Attention Patches*.

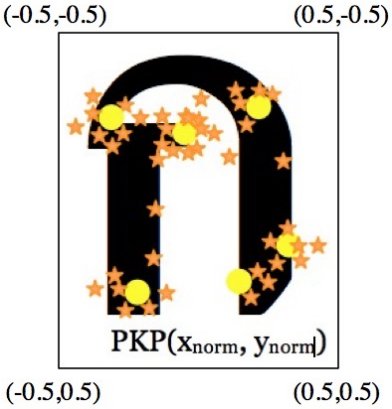


Figure 2.6: Object Attention Patch with SIFT keypoints in a 2D spatial layout.

2.2.5 Recognition method

The SIFT matching algorithm was performed as the basis to classify the testing images. However, the other modified SIFT-based methods were also performed to find the differences in the results. We then modified the SIFT matching functions by combining them with the Region of Interest (ROI), Grid regions, and the PKP's location. Algorithm 2.3 shows the recognition procedure that is used in this study. By, input: set of testing images and set of models, Img_m = Img_1, Img_2, .., Img_m; $Model_n$ = $Model_1$, $Model_2$, .., $Model_n$. Where m = images (1 to m); n = models (1 to n); $R1$ = results of matching by descriptor; $R2$ = results of matching by location.

Algorithm 2.3 Recognition

```
1: for Img_1 to Img_n do
2:     for Model_1 to Model_n do
3:         for pkp_1 to pkp_r do
4:             R1_{1..s} ← matchByDesc { Model_pkp, Img_kp }
           end for
5:         for R1_1 to R1_s do
6:             R2_{1..t} ← matchByLoc { Model_pkp, R1_kp }
           end for
       end for
7:     FinalResult ← maxMatchKp { R2 }
   end for
```

The SIFT matching algorithm using SIFT keypoint descriptors is described in

Figure 2.7. Given A and C are the keypoints of the test image, while B and D are the PKPs of the model. B and D contain similar descriptors to A and C, respectively. Descriptor A should be matched to PKP B. However, C should not be matched to PKP D because of a different location. Using the standard SIFT matching technique, it excludes the keypoint's location in the matching process. Descriptor A will be matched to PKP B, and C will be matched to PKP D, which, the result is incorrect matching on C and D according to the aforementioned. Therefore, recognition uses simple descriptor does not sufficient to provide proper matching results.

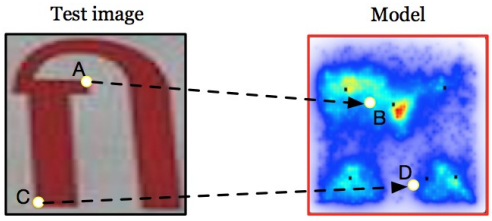


Figure 2.7: Example of SIFT matching algorithm using SIFT keypoint descriptors.

The SIFT matching algorithm combining region of interest (ROI) is taken to solve the location problem from the previous process, as illustrated in Figure 2.8. At the model, the small black dots are the centroid of the group of PKPs, and the ROI is the area in the red circle (drawing the radius measured from the centroid). Investigating the similarity keypoint between A and B also C and D, A is not matched to B, because B is located in the ambiguous area, then it makes the result of recognition is incorrect compared to the assumption. In contrast, C is not matched to D, because D is located in different ROI of C. Therefore, the result of C-D matching is correct. Nevertheless, this method still causes the overlap of the circumference of each ROI, and the matched descriptors sometimes are found outside the ROI, which, produces an error in the matching process.

The SIFT matching algorithm combining grid regions is used to fix the overlap of the circumference of each ROI, and the matched descriptors outside the ROI as demonstrated in Figure 2.9. Define A and C are the keypoint descriptor of the test image, while B and D are the PKPs of the model. The model and test image are divided into 3×3 grid lines. Exploring the resemblance between A and B as well as C and D, A is not matched to B, which B is located in a different grid. The result of

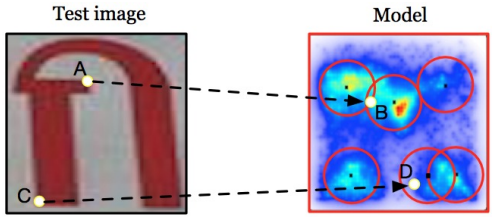


Figure 2.8: Example of SIFT matching algorithm combining region of interest (ROI).

recognition is then incorrect (based on assumption). Meanwhile, C is not matched to D, which D is placed in a different grid, so, the recognition result is correct.

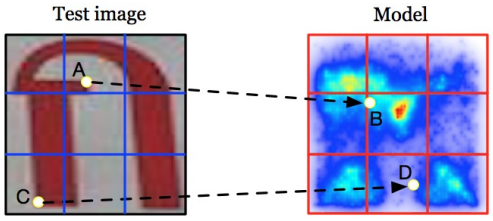


Figure 2.9: Example of SIFT matching algorithm combining grid regions.

According to three experiments, we end up with the propose of the matching technique based on both the descriptor and location, as well as the model's POIs depending on the mentioned functions (Algorithm 2.3). The POI is a centroid of the group of PKPs, which contain PKP's location. All PKPs are members of the group. Therefore, it eliminates the error of the ambiguous or overlap area of ROIs. The POI is also calculated by k-means algorithm, so the region of the group is flexible based on the k-means calculation. We counted the number of matched keypoints to determine the final result. Figure 2.10 explains the proposed method, let A and C are the keypoint descriptor of the test image, while B and D are the PKPs of the model. The small black dots are the centroid of the PKPs group within the model. The red line shows the boundary of each region. The recognition result of A matched to B and the centroid of B is closer to A than other centroids (when all centroid are imitated to test image), is correct matching. While C is not matched to D because of the location of C dissimilar to the area of centroid D. Therefore, the matching is correct. Finally, the matching results give a better performance than the other three methods.

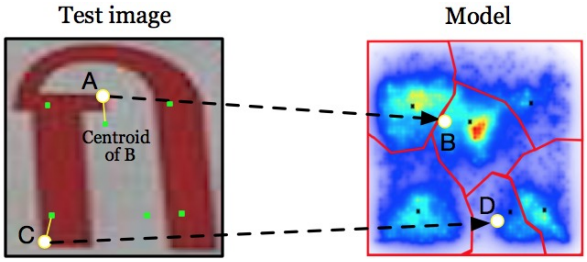


Figure 2.10: Example character images of the TSIB dataset.

2.3 Experiments

2.3.1 Datasets

The Thai image dataset is named the Thai Scene Image dataset by the author (or TSIB in short). The TSIB images are taken by smartphone under different conditions including angle, distance and lighting conditions. Most images are captured at 1,280×720 pixels of resolution. This dataset contains more than 1,500 images in total. This number of the source images provides more than 16,000 Thai characters, which cover all the 10 Arabic numerals, 44 consonants, 21 vowels, 4 tones and another symbols of the Thai language. We took 743 images randomly to be the first version of our dataset. This preliminary dataset is composed of 25 consonants and 13 vowels (8,074 and 3,683 character image samples respectively), which often occur in Thai sentences. Some example characters of the TSIB dataset are shown in Figure 2.11.

Figure 2.11: Example character images of the TSIB dataset.

There are other datasets employed in this chapter. The Robust Reading Dataset

from ICDAR2003 (Lucas et al. 2005) and the Chars74K (de Campos et al. 2009) datasets are selected to be tested as scene images of the English scripts. They contain a total of 17,794 samples of English characters. Based on these numbers of the scene datasets, we can evaluate our proposed model by applying it to both English and an Asian language. Moreover, the Thai OCR Corpus from The National Electronics and Computer Technology Center (NECTEC) [1] that consists of 46 character classes of Thai typed letters is used. Although, these are not scene images, but they are usable in this chapter for evaluating our model in an ideal situation.

2.3.2 The differences between Thai and European scripts

General introduction on Thai script in history, e.g., character-based, character in lines, zones are described. Text detection in natural scene image research is interesting and challenging in Thailand. This is a great opportunity to develop new knowledge. A continuous string of character, although block-segmentable, poses the sensitive problem as occurs in continuous speech recognition and free style written text. A language model is needed to help segmentation such as Markov Models.

- Diacritics and accents English sentences are punctuated with dot punctuation marks and all the words in a sentence with space are then able to detect the character and the recognition process immediately. Even though English has no tone accents in a word as Thai script but some lowercases in English have a dot on the top (i, j), and some letters have a line drawn up/down (f, h, k, l, t, p, q, y, g).

- Word and character segmentation The most challenged of the Thai text is that Thai sentences do not have spaces between the words. The space is only used to separate sentences, or words and numbers or symbols. In addition, Thai alphabet structures are different from other languages scripts. Thai characters are complex geometric shapes, including lines (vertical/horizontal /tilt), circles, and curves. Some letters have line crossings and points of branching. Moreover, some of them look very similar to each other such as shown in Figure 2.12. Finally, a Thai word is composed of consonants, vowels, and tones

[1] http://www.nectec.or.th

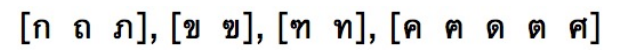


Figure 2.12: Example of the Thai similarity characters.

that can be four vertical zones levels in the written line. Then, it raises the question of how to find the letters in the word. There are some solutions have been used to detect the letters in a word such as top-down: language-based expectancy (e.g., word biographical frequency) and bottom-up: image-based (over) segmentation. Therefore, Thai scripts are rather complicated to find the words in a sentence than English script.

2.3.3 The evaluation of the bag of visual words model

To evaluate the performance of the model, we design the assessment method by taking the model into recognition testing. The model is tested on the recognition of individual characters in every class of character images. We input a single character image into the recognition process using the created model. The result of the recognition shows the class of the character. After receiving all results of the testing procedure, the accuracy rate of the recognition will be calculated each class by counting the correct recognized character compared to the total number of testing images in the class (as demonstrated in Figure 2.14). Eventually, the accuracy rate of all classes is then averaged to get the final accuracy rate of the proposed model. We assume that if the model provides high accuracy in recognition, it should perform the text detection correctly. The experimental set up as follows.

- $The\,Number\,of\,classes$: We selected the class of character images, which contains more than 100 training samples to be tested for TSIB dataset. We use 38 classes of TSIB, 52 classes of ICDAR2003 plus Chars74K divided into upper and lower case and 46 classes of Thai NECTEC datasets.
- $The\,Number\,of\,model\,features$: We tested the character recognition to find the accuracy rates and confusion matrices of the model based on different amounts of the model's features: 300, 500, 800, 1,000, 1,500, 2,000 and 3,000.

The performance evaluation starts with randomly separating source images into two groups in the proportion of 9:1 for training and testing sets. The training set

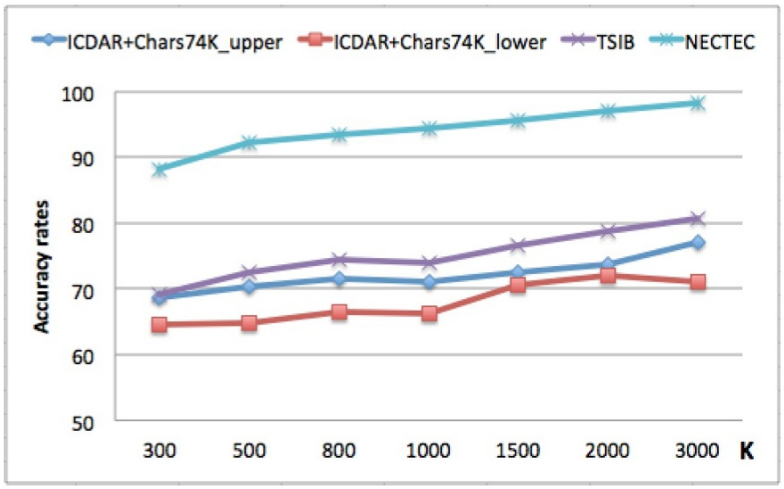


Figure 2.13: Character recognition results for different values of codebook size k (i.e., number of PKPs).

goes to modeling methods and produces different numbers of the model's features: 300, 500, 800, 1,000, 1,500, 2,000 and 3,000. We do a scan matching the descriptor with the distance ratio threshold = 0.6, 0.7, 0.8, and 0.92. In literature, a value of the distance ratio equal to 0.8 is suggested (Lowe 2004) as the optimal ratio, but the ratio = 0.92 provided better results in handling shape variations. The number of correctly matched characters will be calculated as a percentage, representing the classification accuracy. The accuracy rates for different datasets are plotted in Figure 2.13. The results show that a higher number of PKPs improves the accuracy, although at the cost of memory.

Figure 2.14 shows the confusion matrices for a subset of similar-shaped characters, giving an indication of the classifier performance. In Figure 2.14a, the ordinary SIFT matching classifies testing the images of classes 0E19 to 0E1A. The classification result of class 0E19, 0E21, and other classes are 2, 44, 8 and 21 respectively. The wrong classification is significantly decreased ($\chi^2 = 74.3$ and $p < 0.0000001$) when we classify them using SIFT with an attention patch (Figure 2.14b). The overall results for the attention patch approach are given in Table 2.1. In summary, the performance results show our models perform acceptably in recognition with an accuracy rate of more than 70% of all the datasets. This accuracy can be increased depending on the

SIFT

Testing sets \ Results	0E1A	0E1B	0E02	0E19	0E21	0E22	0E29	Others	Error
0E1A	8			6				16	22
0E1B	1	10	2	2	1	1		6	13
0E02			10	1				9	10
0E19	2			44	8	1		21	32
0E21				5	26		2	15	22
0E22			1			24		13	14
0E29							8	3	3

(a)

Our proposed method (the attentional patch approach)

Testing sets \ Results	0E1A	0E1B	0E02	0E19	0E21	0E22	0E29	Others	Error
0E1A	26			0	1	1		2	4
0E1B	2	20				1		0	3
0E02			14			1		5	6
0E19	2	1	1	67	0	2		3	9
0E21	2	1		1	43			1	5
0E22						38		0	0
0E29	1						10	0	1

(b)

Figure 2.14: The confusion matrices of character recognition using the ordinary SIFT classification (a) compare to our approach (b).

Table 2.1: Classification results in percentage.

Datasets	ICDAR + Chars74K WEST upper	ICDAR + Chars74K WEST lower	TSIB Thai	NECTEC Thai (typed)
Classes	26	26	38	46
Samples	1,083	716	1,191	2,162
SIFT	41.00%	41.34%	41.68%	92.46%
SIFT+ROI	64.91%	63.55%	68.26%	97.18%
SIFTGrid	65.19%	64.07%	71.12%	97.69%
Our method	77.10%	74.93%	80.67%	98.29%

codebook size. The similar-shaped characters are better classified substantially. For these reasons, our proposed model should be accurate enough to be used for text detection purposes.

2.3.4 Text detection using object attention patches

We have described the modeling procedure as well as the evaluation of the models in the recognition purpose. In this section, we present the preliminary results of the applying of our models to the text detection based on the assumption that our model should be able to patch a character's attention and locate texts in the scene image. The images from the TSIB dataset are selected to be tested, and the detection method is summarized as follows.

The detection starts with the keypoints extracted from the testing image and then matched with the model's PKPs of the characters we have created. After matching, all the matched keypoints are put into the list. For each matched keypoint, we assume that the keypoint is surrounded by some neighbors that are potentially a 'child' of the same character. The neighborhood definition is based on a minimal and maximal radius and a minimum number of KPs in that neighborhood. The number of neighbors and the radius of the area varies depending on the image size. The keypoints that are not in this criteria will be eliminated. Figure 2.15 gives an example of the detection results for a line of Thai text.

(a) (b)

Figure 2.15: Extraction of characters from a background using object attention patches. (a) Original image. (b) Extraction of characters using character model attention patches.

2.4 Discussion and conclusions

2.4.1 Discussion

We have presented a 'textuality' detector using mesoscale attentional patches in natural scene images containing text. The results are very promising for both recogni-

tion and detection. However, some issues need to be discussed.

First, creating a character model must have a sufficient number of (SIFT) keypoints. If there are not enough raw keypoints from the training images, it is important to enlarge the number of keypoints. Some optional methods such as duplicating black and white or other image perturbation will be useful. Second, the number of PKPs directly affects the efficiency of detection and recognition. However, the larger the codebook, the more intensive is the computation and memory consumption. The challenge is to construct reliable small codebooks based on a large representative data set of SIFT KPs. Third, there are many parameters assigned during the model creation, e.g., number of clusters, matched distance ratio threshold and number of POIs that are not absolutely determined yet. These optimal values need further experiments. Fourth, in the current modeling, the scale and orientation of KPs are ignored. It is possible that useful information is lost in this manner. Future work will address this issue.

Finally, the accuracy rates of approximately 70-80% are satisfactory for the character classification in scene images, but it is relatively low when compared to the machine-printed paper text images at 98.29%. That may be because the PKPs are created from different images. If images quality were improved in the pre-processing, the performance would be increased. Eventually, this efficient model could improve detecting and recognizing texts more precisely in scene images.

2.4.2 Conclusion

This chapter has presented the SIFT-based modeling of character objects for scene-text detection and recognition. The construction of models (attentional patches) from natural scenes has been described. The evaluation of character recognition and a preliminary test for text detection shows our proposed model is usable for scene-text detection and recognition purposes. For future work, an algorithm to increase text detection performance is necessary. On the basis of the current framework, there is a potential both in the improvement of the feature scheme for recognition but also for the development of, e.g., NN classifiers that use the current framework as a textuality detector.

Part II

Modeling text, non-text and attention patches

This chapter is an adaptation of paper:

Sriman, B. and Schomaker, L – *"Explicit foreground and background modeling in the classification of text blocks in scene images,"* 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pp. 755-759, 2015.

Chapter 3

Foreground and background classification using text and non-text attributes

Abstract

Achieving a high level of accuracy when classifying foregrounds and backgrounds is an interesting challenge in the field of scene image analysis because of the wide range of illumination, complex backgrounds, and scale changes. However, although classifying the foreground and background using the bag-of-features model gives a good result, the performance of the classifier depends on the designed features. Therefore, this chapter presents an alternative classification method based on three categories of object-attribute features, namely the object description, color distribution and gradient strength. Each feature is computed into a classifier model. The robustness of this method has been tested on the ICDAR2015 dataset. The experimental results show that the performance of the proposed method performs competitively against the results of existing methods in terms of precision and recall.

3.1 Introduction

Mobile devices are the main equipment used for producing digital content such as images, sound, and video. The quality of images captured by mobile devices is much higher than previously. They also embed textual information, e.g., signs or license plates in complex scenes. The information available from classifying these objects is useful for a variety of purposes. However, the object classification in the captured images requires a proper foreground and background region separation. For the textual information in a real-world scene, it is hard to predict

multiple text patterns, orientations or colors against a background because of the cluttered background. In addition, scene images are often captured under various lighting conditions and viewing angles. Unlike face detection, which is unalterably composed of eyes, nose, and mouth, text detection is rather difficult because all different words consist of uncertain character combinations. Therefore, text and non-text classification in a scene image is currently particularly challenging and needs further study.

Figure 3.1: Examples of the ICDAR 2015 dataset. (a) Text (foreground) blocks. (b) Non-text (background) blocks.

Foreground (FG) and background (BG) classification results in the literature are still not optimal (Wu et al. 2008, Alves and Hashimoto 2010) due to a diversity of the object in a natural scene (Figure 3.1). Hence, a new attempt that considers FG and BG as object classes would be useful for their classification. Additionally, in computer vision, an intelligent system that can predict and answer the question, "Is this a text or non-next block?" encourages correct text detection, extraction and character recognition. However, the success of classification techniques is mostly confined to printed text documents (Su et al. 2013) with most homogeneous image backgrounds.

The classification technique using the bag-of-features model has yielded promising results in the research of image analysis and classification. Some bag-of-features derived methods, for example, the use of color information for human action classification and detection in static images (Khan et al. 2013), have obtained good results. Alternatively, the approach of using multiresolution representation into a bag-of-features model (Zhou et al. 2013) also represents a successful classification of objects

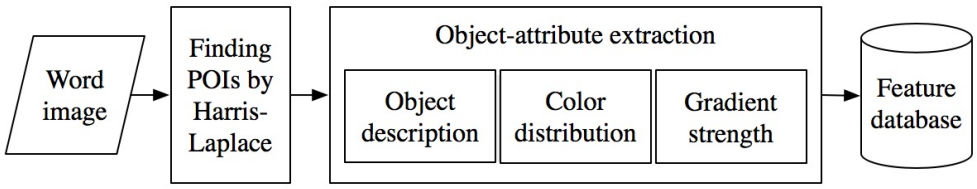


Figure 3.2: Overview of the feature extraction process for text/non-text regions.

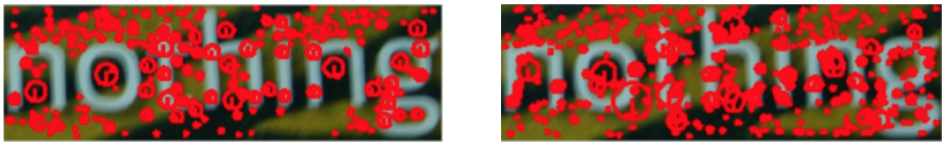
in scenes. However, the aforementioned techniques have completed human action and scene classification, unlike the characteristics of FG and BG.

In this chapter, we propose an alternative approach for classifying text and background by first defining the common characteristics of the FG and BG in a scene image. Next, we introduce a novel classification technique based on the BOFs involving the object description, color distribution, and gradient strength. This top-down scheme is utilized to create absolute text and non-text feature models that can be used to classify the FG/BG regions.

3.2 Object-attributes extraction

All text (FG) and non-text (BG) zones are considered as objects in the scene image. Their attributes can be used to generate an object model, namely 'codebook' or 'Bag-of-visual-words (BOVW)'. Figure 3.2 illustrates the feature extraction process for text/non-text regions. The first stage, objects (text/non-text) in a scene image, vary in scale, orientation, etc. A feature detector such as SIFT (Lowe 2004) is particularly designed for geometric invariants based on the point of interests (POIs) in the picture. It can be used to find the object descriptions. However, detecting the POI using SIFT (DoG algorithm) sometimes provides insufficient keypoints. The lack of POIs would constitute a major limitation in obtaining higher-level object attributes, e.g., descriptions, color, and a strong gradient. The more POIs there are, the richer the description of the object attributes. Therefore, turning to the Harris-Laplace (HL) detection method (Mikolajczyk and Schmi 2001) provides more number of POIs than the DoG in SIFT, as shown in Figure 3.3. Hence, the individual FG and BG zones are detected by the prior text localization.

Next, the characteristics of objects in images: object description, color distribu-



(a) POIs by DoG (217 points) (b) POIs by Harris-Laplace (HL) (421 points)

Figure 3.3: The number of POIs detected by DoG (a) and HL (b).

tion, and gradient strength, are taken into account. The three object attribute types [1] are created as features of text and non-text blocks (in Figure 3.4) and put into the feature database. All three characteristics of objects are described as follows.

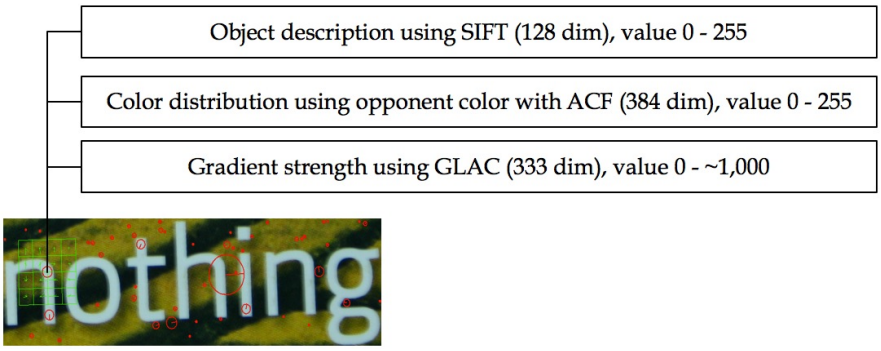


Figure 3.4: Feature extraction at each keypoint (finding POI by Harris Laplace) using three object attributes: object description (SIFT feature), color distribution (opponent color with ACF histogram), and gradient strength (GLAC feature).

3.2.1 Object description

There are various differences in the descriptions between the foreground and background of a scene image. First, text characters are often flat, in 3D, in heterogeneous styles, or are tilted or shadowed. Second, objects in a scene image can be viewed from different angles, which makes the object deformed. Third, the scale of the object is also important for distinguishing text in a natural scene. Finally, the BG has

[1] We use the term *attribute* as a general description of the characteristics of objects in images, and the term *feature* for the computable feature vector

a diverse combination of attributes such as varying patterns, diverse surfaces, and complex colors corresponding to physical aspects present within the scene.

The SIFT algorithm performs very well in terms of scale, orientation and light variation in the natural scene image (Wu et al. 2013). SIFT is therefore selected for extracting an object feature in the first elaboration. The SIFT features consist of a coordinate (x, y), scale, orientation and 128 descriptors. The descriptors are computed using the gradient magnitude and orientation around the keypoint. Several parameters, e.g., steps per scale octave or initial Gaussian blur level (sigma), can be applied to the algorithm to obtain a robust keypoint. As a result, the SIFT algorithm should be suitable for generating text and non-text features of the training zones. Figure 3.5 shows an example of SIFT keypoints extracted from an image using Vlfeat: `http://www.vlfeat.org/overview/sift.html`.

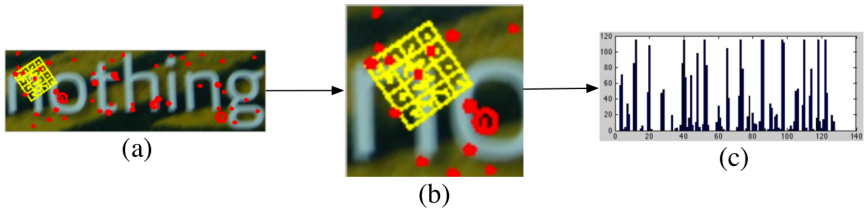


Figure 3.5: An example of extracting a text image feature using HL in order to find the POIs, (a) the red circle refers to the POIs. (b) An example of POI features extracted by SIFT comprises of coordination, scale, orientation and an adjacent 16×16 region (a region is 4×4 sub-regions). (c) The histogram of 128 dimensional feature descriptions of this keypoint.

3.2.2 Color distribution

A real-world scene image is full of colors, lighting conditions and shades. However, the color of the text in natural scenes is regularly distinct from the surrounding area. Many characters in a text line usually have similar colors. Conversely, the color distribution in a background is typically more complicated than the text because several objects are mixed together. Hence, the color distribution could be utilized to separate the FG and BG.

Color descriptors can be used to represent the color distribution of text and non-text blocks. The image is normally captured in color space, and according to these assumptions, the histogram of the colors in the BG should be different from the FG (text). Second, the text color might be rather homogeneous while BG might be more heterogeneous. Finally, images may contain shadows or other irrelevant intensity fields. To extract these mentioned descriptions, the images colors could be separated by the opponent color space into three channels. Channel O_1 and O_2 represent the color information, while O_3 represents the intensity information in Eq. (3.1) (Hurvich and Jameson 1957).

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} (R-G)/\sqrt{2} \\ (R+G-2B)/\sqrt{6} \\ (R+G+B)/\sqrt{3} \end{pmatrix} \tag{3.1}$$

Determining the color distribution of an image can make use of the histogram of the autocorrelation function (ACF) (Box and Jenkins 1994). Using the ACF allows us to identify an appropriate time series between two consecutive values, x_t and x_{t+k}. The time series may be implied by x_1, x_2, ..., x_t, given that x refers to the value and t refers to the time period. This time series is called the autocorrelation of lag k, where k = 1, 2, ..., 256. The lag k of the ACF is defined by Eq. (3.2) where $\overline{x} = (\sum_1^n x_t)/n$.

$$\widehat{\rho}(k) = \frac{\sum_{t=1}^{n-k}(x_t - \overline{x})(x_{t+k} - \overline{x})}{\sum_{t=1}^{n}(x_t - \overline{x})^2} \tag{3.2}$$

Therefore, the color distribution features are expected to be extracted from each color channel. The POIs in the picture are cropped to 20×20 pixels expanding their regions from the center. The cropped area is used to produce the histogram and calculate the ACF, which normally provides 256 dimensions. In order to keep the number of dimensions equal to 128 dimensions which is the same as the SIFT, the odd positions are selected from 256 dimensions. As a result, 128 dimensions of the ACF remain in each color channel and they are combined together. The feature vector ultimately consists of 384 dimensions, as depicted in Figure 3.6. Finally, all the raw features of the text and non-text zones are stored in the feature database to

be clustered in the next step.

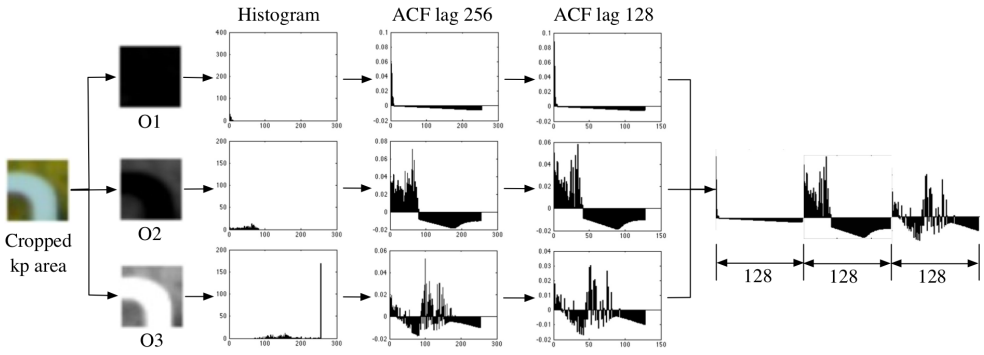


Figure 3.6: The color distribution feature based on opponent color space.

3.2.3 Gradient strength

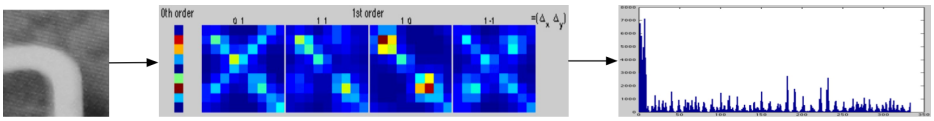
The text gradient from a printed document (dark ink and white background) is unlike the text in a real scene, which contains many colors on a cluttered background. Using color gradients and intensity adjustment techniques can make it easier to find the gradient strength level of objects (Ezaki et al. 2004). Therefore, it is possible to use a strong gradient level to indicate the boundaries of FG and BG.

The intensity of the strong gradient of the object could distinguish FG from BG. Focusing on the pattern of a characters edges in a scene image, it usually remains a similar shape though the font styles are different. Unlike the edges of the objects in the background, their patterns are typically uncertain and no specific shape. To describe edge alignments, a shift-invariant type and a local image descriptor type (Kobayashi and Otsu 2008) is a potential technique. This method is based on spatial and orientational auto-correlations of local image gradients (Gradient Local Auto-Correlation or GLAC). The image gradients are described in terms of their magnitude and orientation. So, the level of gradient strength is computed by the GLAC formulation as shown in Eq. (3.3).

$$0^{th}order \quad R_{N=0}(d_0) = \sum_{r\epsilon I} n(r) f_{d_0}(r)$$

$$1^{st}order \quad R_{N=1}(d_0, d_1, a_1) = \sum_{r\epsilon I} min[n(r), n(r+a_1)] f_{d_0}(r) f_{d_1}(r+a_1) \quad (3.3)$$

Let I be an image region and r=$(x, y)^t$ be a position vector in I. The gradient orientation vector (G-O vector) is represented by $f(\epsilon R^D)$, where orientation $D = arctan(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$. The order of autocorrelation $N\epsilon$ {0,1}, the gradient magnitude $n = \sqrt{\frac{\partial I^2}{\partial x} + \frac{\partial I^2}{\partial y}}$, $a_{1_{x,y}}\epsilon\{\pm\Delta r, 0\}$, a (scalar) weighting function $\equiv min(\cdot)$ and f_d is the d-th element of f. Figure 3.7 shows an example of extracting a feature by GLAC.



(a) (b) (c)

Figure 3.7: The visualization of features extracted by GLAC. (a) The gray cropped keypoint area. (b) An example of features extracted by GLAC of the 0^{th}order with the number of gradient orientation bins (D) set to 9 (small bar) adjoined with the 1^{st}order referring to the joint distribution of orientation pairs of local gradients in 9×9 bins. (c) The histogram of the GLAC feature consists of 333 dimensions (D + $4D^2$).

3.3 Distance methods

The simplest way to analyze the data classification or the similarity of two objects is to find the distance between them. The distance methods can be calculated with two quantitative variables. If the result is very small, then this may infer that the two objects are similar. In addition, the distance method is also used to make data clustering more accurate. Hence, there are six distance algorithms involved in the experiments.

3.3.1 Cosine distance

This is used to compute the distance between two vectors by looking at the angle between the vectors. Therefore, the magnitude or weight is not taken into account. The cosine can be calculated by 1 minus the action dot product of two positive vectors a = (a_1, ... , a_n) and b = (b_1, ... , b_n), the cosine formula is defined by Eq.(3.4).

$$d_{ab} = \frac{a \cdot b}{\|a\| \, \|b\|} = 1 - \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{3.4}$$

3.3.2 Correlation distance

This is the statistical scale that measures the data dependency of two values or vectors. The correlation distance as shown in Eq.(3.5) is measured by the variance and standard deviation, which values are between 0 and 1.

$$d_{ab} = 1 - \frac{cov(a,b)}{std(a) * std(b)} = 1 - \frac{(a-\bar{a})(b-\bar{b})^{'}}{\sqrt{(a-\bar{a})(a-\bar{a})^{'}} \sqrt{(b-\bar{b})(b-\bar{b})^{'}}} \tag{3.5}$$

3.3.3 Euclidean distance

This is a basic distance measure which is used to compute the distance between two points. It is very popular in many types of work because it is easy to understand and the calculation is similar to using a ruler to actually measure the distance. Eq.(3.6) describes the computing distance between points a and b.

$$d_{ab} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{3.6}$$

3.3.4 City block distance

This algorithm is also known as the Manhattan distance. It measures the distance between 2 cases, for instance, it demonstrates the distance between two points in a

city road grid. It considers the sum of the absolute difference of their coordinates with the same unit, as illustrated in Eq.(3.7).

$$d_{ab} = \sum_{j=1}^{n} |x_{aj} - x_{bj}| \tag{3.7}$$

3.3.5 Spearman distance

This is a statistical measure, which is used to estimate the strength of a linear relationship between two ordinal vectors. The calculation distance of the vectors by the Spearman distance in Eq.(3.8) is more special than other statistics; first the vectors are ranked and compared with the data sets to get $\sum d^2$. Then the values are inserted into the simplified Spearmans Rank Correlation Coefficient formula and n is replaced with the total number of ordinal features.

$$d = 1 - \frac{6 \sum d^2}{n(n^2 - 1)} \tag{3.8}$$

If there were ties in any of the previous steps, the standard Spearman's Rank Correlation Coefficient formula in Eq.(3.9) is used instead.

$$d = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2 \sum_i (b_i - \bar{b})^2}} \tag{3.9}$$

3.3.6 Chebychev distance

This is also known as the chessboard distance, in which all 8 adjacent cells from the given point can be reached by one unit. It calculates the distance between two vectors or points by selecting the maximum absolute difference between the coordinates of two vector elements. Eq.(3.10) demonstrates how to compute the Chebychev distance between vectors a and b.

$$d_{ab} = max_i \{|a_i - b_i|\} \tag{3.10}$$

3.4 Text and non-text classifier based on the bag of visual words technique

3.4.1 Codebook creation

All the features are extracted from images in Section 3.2; they are typically different regarding their positions (coordinates (x, y)), but the descriptors could be similar. The similarity of the descriptor may exist when the keypoints are generated from similar parts of objects. Therefore, text and non-text attributes are utilized as raw data to establish object models (or codebooks). Then a codebook approach is used for each attribute type using clustering. Feature clustering based on Cluster Analysis is crucial to group descriptors in order to allocate into the same group. Using the simplest intuitive learning algorithm such as k-means is a Nonhierarchical Cluster Analysis or partitioning clusters. This algorithm can be calculated faster than hierarchical clustering because it does not compute the proximity matrix in order to distinguish objects. The k-means algorithm will split the object into k groups, or the partitions are decided by the experience.

To create the codebook of object description (SIFT), color distribution (CDis), and gradient strength (GLAC), according to the proving of k by Sriman and Schomaker (Sriman and Schomaker 2015b), the large number (k = 3,000) of clusters provided high accuracy, but the computational time was also increased due to the high complexity. Based on that, the descriptors are clustered into k = 4,000, namely the prototypical keypoints (PKP) for both FG and BG classes, where the computational time remains acceptable. These PKPs which are used for producing the codebook that consists of 8,000 PKPs for each attribute. The first half represents the characteristics of the text object, whereas the second half represents the background object, as shown in Figure 3.8.

It can be noted that there are different numbers of keypoints extracted by SIFT, CDis, and GLAC, which are derived from the POIs that are detected by HL. The CDis and GLAC features are obtained by cropping small areas (20×20 pixels) at each POI. The cropped areas are then extracted its features by CDis and GLAC algorithms. While, some POIs cannot provide keypoints by features extraction method

using SIFT. Therefore, the number of extracted keypoints by SIFT are less than CDis and GLAC algorithms.

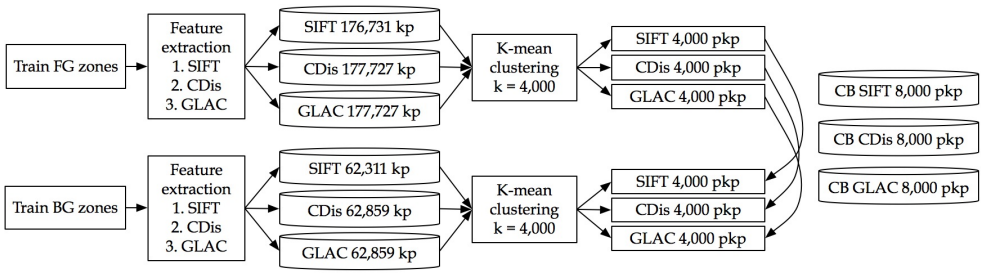


Figure 3.8: Codebook construction for object description (SIFT), color distribution (CDis), and gradient strength (GLAC).

3.4.2 Computing optimal distances for the codebooks

The assumption of classifying the keypoints of the candidate block is that the candidate keypoint will be matched to the PKP of the codebook. If the matched PKP is located in the first half of the codebook, it should be the keypoint of text as previously mentioned. The matching between the keypoint of the train/test sets and the PKP is performed by calculating the distance using six algorithms: Euclidean, city block, Chebychev, cosine, correlation, and Spearman. To get an optimal matching algorithm and reliable results, the experiments are set up in four groups, which consist of the random train and test keypoints (5:5, 6:4, 7:3 and 8:2). The experiments are repeated five times in each group.

Table 3.1: The percentage of text and non-text keypoints classification matched to codebooks in different distance algorithms.

Distance methods	Sift		CDis		GLAC	
	Train	Test	Train	Test	Train	Test
Cosine	98.12	69.64	93.75	78.23	84.56	70.52
Correlation	98.02	70.57	92.65	76.31	84.52	70.52
Euclidean	98.13	69.37	93.88	77.66	84.65	69.83
City block	97.62	72.38	90.58	75.73	85.69	70.50
Spearman	92.70	66.83	88.65	76.53	70.41	62.56
Chebychev	95.17	61.75	84.07	71.45	79.59	67.91

Table 3.1 depicts that distances yield different results (χ^2 = 210.15, df = 4 and $p < 0.0001$). The train and test are different (χ^2 = 3150.21, df = 1 and $p < 0.0001$). The features are different (χ^2 = 600.11, df = 2 and $p < 0.0001$). The training and test result differences are slightly dependent on the distance used (χ^2 = 12.69, df = 4 and $p < 0.0129$). The different features need different distance functions (χ^2 = 120.07, df = 8 and $p < 0.01$). They also have a different 'collapse of performance' when going from train to test data (χ^2 = 309.98, df = 2 and $p < 0.0001$). Based on the statistical results analyzed by chi-square, for this reason, the model construction of each feature is taken by a different distance algorithm.

However, from Table 3.1, when calculating the average of the matching results between training and testing to the codebooks for each distance algorithm, as shown in Figure 3.9, the city block distance yielded an average higher than the other distance algorithms for the SIFT feature and GLAC feature, while the cosine algorithm gave the highest average for the CDis feature. Therefore, the city block algorithm is suited to generate SIFT and GLAC models. Contrary, the cosine algorithm is appropriated for the CDIS model.

3.4.3 Classifier design

The Support Vector Machine (SVM) (Hastie et al. 2009) is a proven method in solving problems on data classification and pattern recognition. SVM model of the SIFT, CDis and GLAC features are constructed by the matching histogram resulted from matching procedure between training set and codebook. These histograms consist of 8,000 bins. The visualization of the histogram is shown in Figure 3.10. The high intensity of the histograms located on the left side is expected to be the keypoints of the text, whereas the high intensity histograms are generated from the non-text objects mostly located on the right side. Next, the histograms are normalized to a standardized value of zero to one [0,1] and are used as training data for the SVM algorithm by specifying class value 1 for the text and -1 for the non-text.

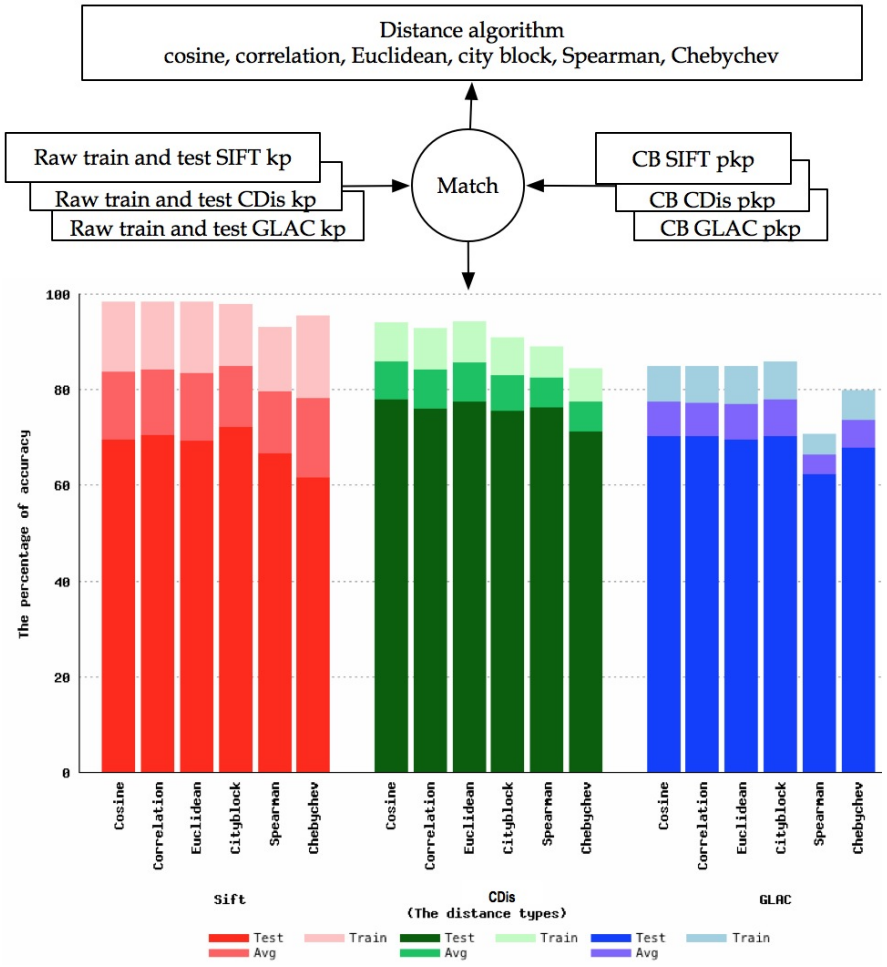


Figure 3.9: Computing optimal distance (Euclidean, city block, Chebychev, cosine, correlation, and Spearman) for object description (SIFT), color distribution (CDis), and gradient strength (GLAC). The graph shows that the city block distance provides a higher average result than the other distance algorithms for the SIFT and GLAC features, which differs from the CDis feature where the cosine distance gave a better result.

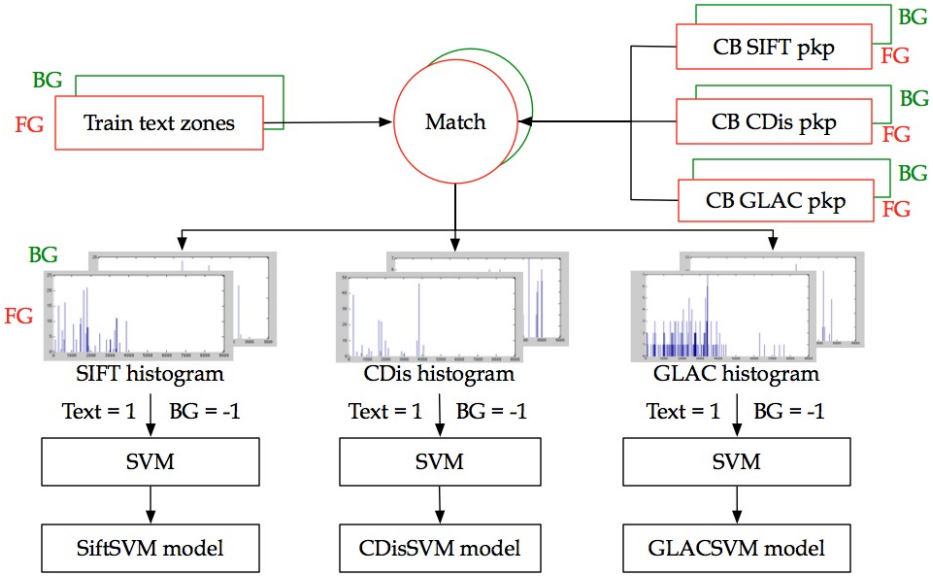


Figure 3.10: Creating an SVM model for SIFT, CDis and GLAC.

3.5 Experiments

We design the experiment to discover the performance of our created classifier based on the classification of text and non-text block images. The classification process is taken to test the ability of the classifier to separate text and non-text images. The testing image is classified by the classifier and report the result as the class of text or non-text. The number of correct classification is accumulated and calculated to the classification rate of the classifier. The dataset and the results show in the following sections.

3.5.1 Dataset

The proposed methods were tested on the ICDAR2015 dataset: `http://rrc.cvc.uab.es/?ch=2&com=introduction`, which contains 229 and 233 images in the train and test sets, respectively, representing 91% of ICDAR2003. For creating the models for the train set, 1,313 FG zones (derived from cropping 849 and 464 random images) and 1,286 random BG zones were obtained. For the testing, 1,095 cropped

FG and 1,035 random BG zones were used to evaluate the classifiers performance.

3.5.2 Results

The SiftSVM, CDisSVM, and GLACSVM classifiers were individually evaluated on the testing set of text (FG) and non-text (BG) blocks (Table 3.2). The classification rate for text by SiftSVM at 89.59% is higher than the other two classifiers (CDisSVM and GLACSVM classifiers). Meanwhile, the consideration of the color distribution using the CDisSVM classifier gives medium accuracy for text at 69.59%. So, color distribution (CDis) could be improved by combining it with another feature. The object descriptor feature using the SIFT concept is selected to be applied to the color feature of the point of interest (POI). Concerning the creation of the new feature, an image (text/non-text) is transformed into three layers of the opponent color space ($O_1O_2O_3$) (Hurvich and Jameson 1957), which provides more efficient color detail than common RGB colors. Each channel of opponent color is extracted by the SIFT descriptor on the POI detected by HL, and then all the descriptors are combined to 384 dimensions (in Figure 3.11). After that all the text and non-text features are clustered to build the codebook using the procedure in Section 3.4.1. Finally, the new codebook named the OppSift is used to create a new classifier called OppSiftSVM (in Section 3.4.3). The classification performance using the OppSiftSVM classifier gives a classification rate significantly increased at 94.34% and 88.31% for text and non-text objects, respectively.

Based on the advantage of the combined features, the two classifiers: OppSiftSVM and GLACSVM are employed to classify the observation feature as text and non-text using the voting weights in Eq.(3.11) and Eq.(3.12). Where $w_{(A)}$ is the weight of the first classifier (Eq.(3.11)), $Acc_{(A)}$ is the accuracy of the first classifier and $Acc_{(B)}$ is the classification result of the second classifier. For $w_{(B)}$ the weight of the second classifier is carried out in the same way as defined by Eq.(3.12).

$$w_{(A)} = \frac{Acc_{(A)}}{(Acc_{(A)} + Acc_{(B)})} \tag{3.11}$$

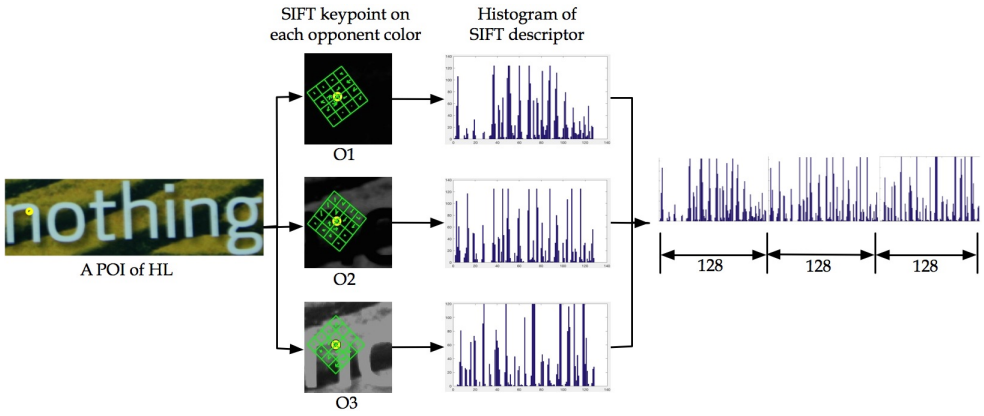


Figure 3.11: OpponentSift feature.

$$w_{(B)} = \frac{Acc_{(B)}}{(Acc_{(A)} + Acc_{(B)})} \tag{3.12}$$

To get final accuracy, weights are applied to the consideration phase of the classification together with the probability ($prob$) resulting from two classifications using Algorithm 3.1. Given $w_{(A)}$, $w_{(B)} \epsilon \mathbb{R}$ are the weight of classes A and B computed by Eq.(3.11) and Eq.(3.12), respectively. The number of FG and BG classifications using a combination of A and B classifiers (AB) is referred to as $finalFG_{(FG,AB)}$ and $finalBG_{(BG,AB)}$ = 0. Further, $prob_{(A)}$ and $prob_{(B)}$ are the probability of the result acquired by the prediction method using the Support Vector Machine (SVM) classifier within the procedure $[class, prob] = predict(SVMModel, candidate)$. This returns a matrix of probability ($prob$), indicating the likelihood of a particular class. In this work, we use $prob_{(A)}$ and $prob_{(B)}$ to represent the likelihood of classes A and B. For the probability result of combining two features, $prob_{(AB)}$ is calculated by Eq.(3.13).

$$prob_{(AB)} = (prob_{(A)} * w_{(A)}) + (prob_{(B)} * w_{(B)}) \tag{3.13}$$

Given $prob_{(A)}$ is the likelihood score of class A, and $w_{(A)}$ is the weight of class A. $Prob_{(A)}$ times $w_{(A)}$ plus the likelihood score of class B ($prob_{(B)}$) times the weight of

Algorithm 3.1 Classifying Testing Image

$finalFG_{(FG,AB)}$ = 0; $finalBG_{(BG,AB)}$ = 0;
$prob_{(A)}$ = 0; $prob_{(B)}$ = 0;
for img_1 to img_n **do**
    $prob_{(A)}$ = predict($SVMModel_{(A)}$,$candidate_{(A)}$);
    $prob_{(B)}$ = predict($SVMModel_{(B)}$,$candidate_{(B)}$);
    $prob_{(AB)}$ = ($prob_{(A)}$*$w_{(A)}$) + ($prob_{(B)}$*$w_{(B)}$);
    if $prob_{(AB)} \cong GT_{(FG)}$ **then**
        $finalFG_{(FG,AB)} = finalFG_{(FG,AB)}$ + 1;
    else
        $finalBG_{(BG,AB)} = finalBG_{(BG,AB)}$ + 1;
    end
end

class B ($w_{(B)}$) provides the final prediction of the candidate ($prob_{(AB)}$). If $prob_{(AB)}$ point to a member of the ground truth of FG ($GT_{(FG)}$), the candidate is assigned to class FG otherwise it is assigned to class BG. The classification result of the combination of the OppSiftSVM and GLACSVM gives 95.62% for text and 88.79% for non-text. Consequently, classifying text and non-text by the OppSiftSVM and GLACSVM improves the accuracy rate, as shown in Table 3.2.

Table 3.2: Classification results of FG and BG using our classifiers.

Model classifier	FG(%)	BG(%)
SiftSVM	89.59	82.22
CDistSVM	69.59	84.25
GLACSVM	87.31	85.70
OppSiftSVM	94.34	88.31
OppSiftSVM+GLACSVM	**95.62**	**88.79**

The precision and recall rates of our proposed classifier are computed and compared to other methods (different datasets and data sizes). The performance of Wus method (Wu et al. 2008), tested with 84 test images, contained 367 text regions (minimum three characters per region) of the ICDAR2003 and showed the recall and precision rates were 76.29% and 78.87%, respectively. On the other hand, the Alvess method (Alves and Hashimoto 2010) showed a better result for precision and recall at 97% and 88%, respectively, with similar datasets. With regard comparing the experiment of the two previous methods to our proposed. We tested the dataset in Section 3.5.1, in which the number of image regions is bigger than the other two

experiments. The final results of precision and recall of our proposed classifier are at 94.89% and 84.17%, respectively (Figure 3.12).

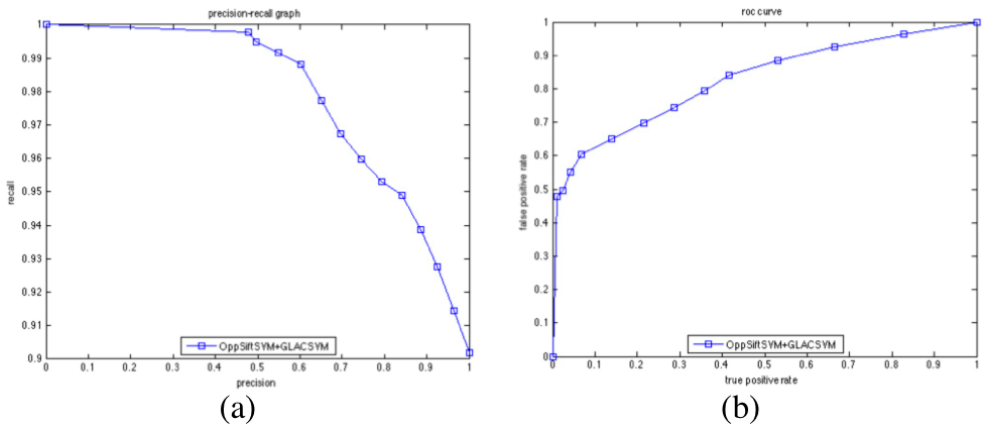


Figure 3.12: (a) Precision and recall (b) ROC of the proposed classifier.

3.6 Conclusion

We proposed an explicit foreground and background modeling in the classification of text blocks in scene images based on extracting object-attribute features. The object features have been defined by three categories: object description, color distribution and gradient strength. Then we created the object models from these categories collaborating with the SVM in order to generate the classifier. The experiments and evaluations with the ICDAR2015 dataset show that our method performs effectively in foreground and background classification in comparison with existing algorithms by the accuracy rate of precision and recall at 94.89% and 84.17%, respectively. In future work, we will manipulate this method to improve the performance of text localization and text extraction in scene images based on the high accuracy of our proposed model of the classifiers.

This chapter is an adaptation of paper:

Sriman, B. and Schomaker, L – *"Multi-script text versus non-text classification of regions in scene images,"* Journal of Visual Communication and Image Representation (JVCI), volume 62, pp. 23-42, 2019.

Chapter 4

Foreground and background classification using multimodal features

Abstract

This chapter continue our analysis of text versus non-text region classification for scene images. Then we propose a feature-based classification of image blocks using the color autocorrelation histogram (CAH) and the well-known scale-invariant feature transform (SIFT) algorithm, yielding a combined scale and color invariant feature vector suitable for scene-text classification. To evaluate this approach, features were extracted from different color spaces such as RGB or YUV, represented in the form of a particular autocorrelation histogram and adjoined with a SIFT descriptor of given keypoints. This approach can be used as an efficient classification model for text in a scene image. Parameter tuning is performed and evaluated in this study. As regards the final classifying methods, the regular nearest neighbor (1NN) and the support vector machine (SVM) were compared. The results revealed that Asian script gives better results than Western script when using 1NN classification. The results for the SVM depend on the model parameters, in particular C determining the level of misclassified results. Finally, the performances of the proposed model appear to be robust and suitable for Asian scripts such as Kannada and Thai, where the scene text fonts are characterized by a high curvature and salient color variations.

4.1 Introduction

Text detection and recognition provides useful enrichment in many computer-vision applications. The function has been advocated as an aid for persons with

a reading impairment or tourists needing automatic translations. It can also provide a means to improve GIS systems, especially for pedestrians, using visual landmark recognition for navigation. This function is also important in Google Streetview for providing street name and house number recognition, and context retrieval on image/video-based applications. In pedestrian navigation support systems, finding a particular path towards the actual final goal within a room radius in a large building such as an office or hospital is becoming increasingly important but is still unsolved. To recognize items in a complex scene image captured by digital cameras or smart devices entails several problems in image processing, due to the variety of color patterns, blur, noise or other image distortions.

Specifically, finding text embedded in photographs of urban/natural scenes is still a difficult pattern recognition problem. This is because there are often complex background images with similar colors and gradients to the target text elements. Contrary to paper-based optical character recognition (OCR) problems where the background consists of an evenly distributed color and non-salient texture, the background image content in scene images is highly complicated. Additionally, there are various text styles (fonts), multiple colors and scale, and perspective variations both in Western and Asian scripts. There is also much more text than just street numbers. Advertisements may be presented as three-dimensional physical text objects in a multitude of shape variants. Texts in a scene image can be viewed from different angles, deforming the observed objects and in addition, shadow and poor lighting conditions have a big influence, if the goal is to capture the text information in a real-world scene. However, despite these difficulties, recognizing the text strings in a real scene provides highly useful, uncomplicated and explicit information on the ambient environment to humans and AI systems. Therefore, classifying text versus non-text in a natural scene is an important goal.

There are various classification techniques to discriminate texts from complicated backgrounds such as artificial neural networks (ANN) (Panhwar et al. 2019), nearest-neighbor (1NN) (Neumann and Matas 2013), support vector machine (SVM) (Kim et al. 2003) and the deep convolutional neural network (DCNN) (Shi et al. 2017). Recently, CNN or deep learning has evolved into a powerful technique used in image analysis, e.g., scene text detection, classifying objects and object recognition. However, there is a complex modeling process; for instance, multi-digit

number recognition from Street View requires an astonishing number of 11 network layers (Goodfellow et al. 2013). Deep learning therefore requires not only a huge amount of labeled sample data but also huge computing resources. Therefore, in order to be less time-consuming, the size of the raw input image from the camera is usually reduced (He et al. 2016). It should be noted that the current successes of CNNs start with the availability of a correctly cropped sub-image of usually approximately 256×256 pixels. This represents the width of 16-32 OCR-legible characters only. In our application, however, the input image will be a complete scene of, e.g., 2-41 megapixels. A full convolution of a 256^2 pixel patch over such a large image is computationally too demanding. Fast and efficient prior detection of relevant image regions will be required in convenient real-life applications. In simple terms, one needs to be able to detect text (FG) and non-text (BG) blocks in a computationally effective and convenient manner, before the stage of text recognition (OCR) itself.

Local, structural shape features such as SIFT and SURF are an important solution for the detection of relevant text regions. Such local features can be computed easily and have been shown to be well suited for matching and recognition as well as for many other applications where occlusion, background clutter, and other content variations occur. To compute points of interest (POI) in the image, it has been found that the Harris-Laplace (HL) detection method provides a larger portion of salient image patches than SIFT (Sriman and Schomaker 2015a). HL-detected POIs identify rich local information allowing higher-level object attributes, e.g., descriptions and color, to be completed for that region. However, whereas HL appears to be more useful for POI (keypoint) detection, the SIFT descriptor may be very useful for additionally describing a region, due to the scale and orientation invariance, if it can effectively discriminate between foreground and background regions.

However, images of a natural scene are typically colorful and have much more texture than scanned printed text documents, which usually contain a white background and dark foreground. This observation seems to imply that some color features, e.g., the histogram of real-valued color features in different color spaces should be used to identify FG/BG properly. In addition, the illumination variation, shadows, and reflections within a natural scene, will influence the performance of such a color-based image classification. For example, our human visual system will compensate for the observed differences in colors of the same object when the object

is struck by light (Troost and Weert 1991), but it is still a challenge to realize color constancy in an algorithm.

When an image is captured, it is converted from analog signals to a digital image, in a quantization process in order to convert the intensity of light to a number. There have been several color intensity representations used in various application fields, since the development of color theory (Goethe 1982, Koenderink 2010). Nevertheless, the most commonly used representation uses 256 intensity levels in the range 0-255 for pixel value for each of the colors red, green and blue. Due to the complicated relation between lighting conditions and reflective properties of text and background material, some transform in the raw RGB values is needed to realize color constancy. To illustrate this, a text sign was captured with a smartphone at different times of the day, i.e., in the morning and evening, as shown in Figure 4.1. We experimented with some feature schemes, which did not produce a promising outcome (Sriman and Schomaker 2015a). The word image at the top was taken on a sunny morning whereas the photo at the bottom was taken in the evening under cloudy conditions. For each photo, the histogram of R|G|B color intensities $I\epsilon[0, 255]$ is computed. The histograms of the images taken during daytime and evening are noticeably different (Figure 4.1). What is needed is a change that retains the useful color information better than RGB histograms, disregarding the irrelevant lighting condition.

Autocorrelation is a form of time series analysis. It describes the correlation between data sequences in a single series of sample values, at several delays in 'time'. Using autocorrelation, the position of salient elements (e.g., peaks) in time is made irrelevant, whereas the information about their shape is retained. Therefore, it was hypothesized that it would be possible to represent color information in an intensity-invariant manner, by using the autocorrelation functions (ACFs) for the histograms of color intensity in red, green and blue, or even in other color-coding schemes. Although ACFs have been used in image processing (Beaudoin and Beauchemin 2002, Matsukawa and Kurita 2010), we believe this is a novel approach. Summarizing, we expect the heuristic of autocorrelation in time series analysis to reduce the illumination problem in FG/BG detection in natural scenes. After applying ACF to the histograms, the graphs of the images taken at different times appear much more similar (Figure 4.2). On the basis of such observations we expect

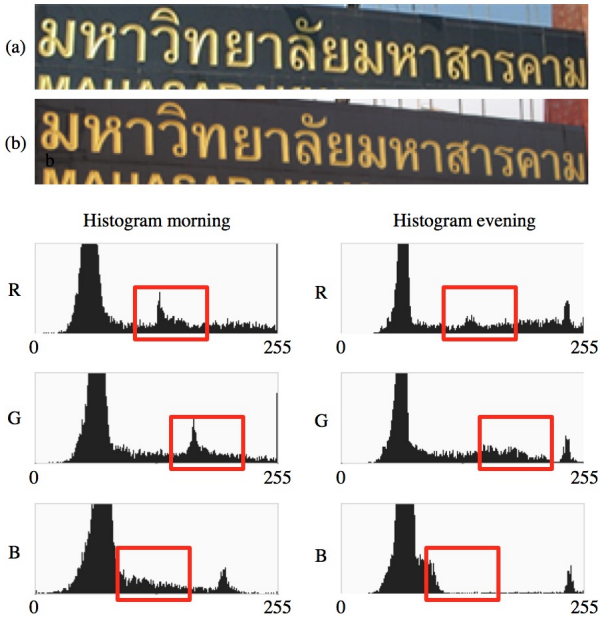


Figure 4.1: Illustrates one text under sunny conditions vs evening. Color histogram in R, G and B for a text object photographed in the morning (a) and in the evening (b). The differences are highlighted in corresponding rectangles on the left and the right.

that the autocorrelation function can be used to attain color constancy, at least to an important extent.

In order to test whether this expectation holds, we present text/non-text region classification, using the image features (i.e., 'color autocorrelation histogram' (CAH)) and compare it to standard SIFT. The basic idea is to obtain the image features around a salient point of interest (POI) by using the Harris-Laplace (HL) detector in order to find the prominent characteristics (e.g., color and descriptor) of the images around a POI, so a region of interest (ROI) can be defined. All the detected regions will be represented by features extracted from this region. So, at each POI, a feature vector is extracted using the global feature (CAH) and local feature (SIFT) in order to represent the region information. Each region can be divided into the foreground (text, FG) class or the background (non-text, BG) class. However, in fact, this dual-class definition is a simplification. Unlike the situation of ordinary

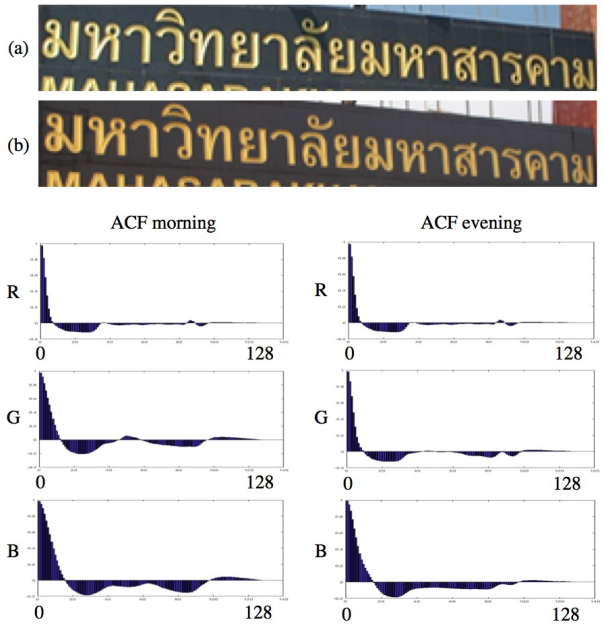


Figure 4.2: Illustrates a text under sunny conditions vs evening.

document analysis, the BG class is a container, in which both simple signs or plate colors are represented as well as complicated natural or urban scenes. Therefore, as we have suggested elsewhere (Sriman and Schomaker 2015a), explicit modeling is required: intensity differences are not a sufficient heuristic for FG/BG separation. These feature vectors should be gathered into categories. The bag-of-word model was first reported to use the frequencies of words from a dictionary for document representation (Salton and McGill 1986). Today, a similar bag-of-visual-words (BOVW) method is often used for scene image processing. It is convenient and effective in many computer vision applications such as action recognition (Sharma et al. 2012, Khan et al. 2013), image classification and character recognition. To obtain the codebooks or BOVW, all feature vectors are grouped into clusters using the k-means algorithm. Due to the visual complexity of scenes, k may be a large number. There are several criteria to be involved, so the best selection could predict the best result.

To find the optimum performance, the experiments are designed as follows:

(1) In order to study the impact of different conditions for CAH based on RGB

color, we explore the optimal parameter values for: i) patch sizes, ii) autocorrelation types, iii) distance types, iv) normalization types, and v) codebook sizes which require a computationally intensive grid search. The best result will then be chosen to create the CAH classifier and SIFT classifier.

(2) There are many kinds of color spaces; therefore, nine color spaces using the selection criteria for creating CAH feature are compared to get the best color classification result.

(3) In the stage of comparing text/non-text classification, we want to test the basic features first, such that e.g., CNN-based methods can be contrasted with this in later research. So, CAH and SIFT features are compared for accuracy by two different model classifier methods such as an ordinary nearest neighbor (1NN) and a proven method such as SVM.

(4) Finally, we attempt to exploit the relative benefits of the CAH and SIFT classifiers by adjoining the feature vectors and evaluating the effect on text/non-text classification.

4.2 Related work

Not all regions in a scene image are equally informative. To identify texts in images, the concept of 'points of interest' plays an important role. The idea is to detect some critical points in the image. These may include corner points, edges and so on. To describe the characteristics of these points, several approaches have been proposed for extracting their features using local descriptors, e.g., SIFT and Harris-Laplace (HL). Azad and et. al. (Azad et al. 2009) presented the features that combine the Harris corner detector with the SIFT descriptor for real-time localization and recognition of textured objects. The proposed approach can construct the features within approximately 20ms computing time by recognizing an image of size 640×480 pixels and localizing a single object at 30Hz of frame rates. Wang et al. (Wang et al. 2009) and Weixing et al. (Weixing et al. 2015) used HL to detect and locate image features for automatic image registration based on the effectiveness of scale-invariance of HL. However, the multi-scale approaches have a common defect. Therefore, based on the scale-space of HL together with the reliability of SIFT descriptors, Zhang et al. (Zhang et al. 2009) improved the algorithm to remove the redundant points de-

tected by the original Harris-Laplace. Gong et al. (Gong et al. 2016) implemented the application to target tracking using the Harris-Laplace corner to localize the target, and the results promise the feasibility of the proposed method and precisely localize the target; so far it has been used to make an intelligent transportation system.

In addition, a color-based method is a crucial technique for text localization, face recognition, as well as object segmentation. Kwok et al. (Kwok et al. 2009) investigated the separation of foreground and background objects based on the selected distribution with the maximum entropy that affects the aerial images of planted fields. The images are transformed from each RGB color to YIQ, YUV, $I_1I_2I_3$, HSI, and HSV color spaces. The result showed that the method was appropriate for segmentation of color images. Nevertheless, it had a bias effect on gray-scale images. Color space is also beneficial for segmenting the image, e.g., using chromatic and achromatic information in XYZ color signals and separately smoothing them through anisotropic diffusion (Lucchese and Mitra 2001). The results of these experiments have confirmed the effectiveness of this approach.

One crucial step after detecting and localizing potential text fragments in images is to determine the possibility of these fragments (so-called text candidates) to be part of actual texts or non-text objects. Block-based classification is one of the frequently-used techniques that have been reported in the literature. Blocks (rectangular image regions) are generated around the point-of-interest candidates before a classification is performed to classify the blocks into text and non-text blocks. Many previous studies have presented a classification technique to verify text blocks in scene images. Jiang et al. (Jiang et al. 2006) proposed verifying a list of connected components (CCs) (derived from a color clustering algorithm) to be texts and non-texts by a 2-stage classification: coarse classification and then precise classification. The coarse classification included a series of cascaded classifiers, considered five features (geometric, shape regularity, edge, stroke and spatial coherence features) and two thresholds to discard non-text CCs. All accepted CCs from the previous step, and then were judged by the precise classification using SVM. The experiment showed that classifying text provided a more explicit result than non-text. The alternative technique for classifying candidate text regions is two-layer classification presented by Zhu et al. (Zhu et al. 2015). The first layer is to compare the similarity score of CC region blocks, which can filter out most repeat backgrounds. The

second layer is an SVM classifier using a histogram of gradients (HOG) descriptor. The experiments showed that the first layer classified non-text as 63% and text as 97.3%. After the second layer was applied for verifying the text classification, the result increased by approximately 0.4%.

Minetto et al. (Minetto et al. 2014) proposed a technique to classify a box of the text-line candidates of a GPS-tagged high-resolution digital photo of a city, which applied the T-HOG descriptor to generate a descriptive feature. The classifier classified the candidates into text and non-text regions. The classifier was constructed with a multi-cell histogram of oriented gradients (HOG) of Dalal and Triggs (Dalal and Triggs 2005). It was used to analyze the differences in font sizes and ignore irrelevant texture inside characters.

The approach of 'characterness cues' presented by Li (Li et al. 2014) is used to measure the unique properties of characters. The feature includes three property cues: stroke width (SW), perceptual divergence (PD), and a histogram of gradients at edges (eHOG), which are represented by region r. The region r is then used to compute the probability of characters and backgrounds by the Naive Bayes model to observe the distribution likelihood of characters and non-characters for each cue. The observation showed that in the PD distribution, characters tended to have a higher contrast than non-characters, which is different to both SW and eHOG. The evaluations of the proposed methods show that SW is applied to indicate characters and non-characters. However, it is more effective when all cues are combined.

Graphics and scene text discrimination in video document analysis are also challenging and interesting problems because of the clarity and separability of graphics and scene text in video frames. Therefore, Xu et al. (Xu et al. 2014) presented a method for classifying graphics texts and scene texts under the hypothesis that graphics texts are arranged at almost the same location and have a uniform color with a plain background, against scene texts that have a non-uniform color and cluttered background. They use the principle of deviation of the movement of the text block for a few seconds to determine whether it is a scene or graphic text. If there is a high deviation, it is a scene text. On the other hand, graphic text usually has a low deviation in position. The result of the proposed method guarantees that the classifying graphics and scene texts are correctly classified. However, there may be a problem with static scene texts that are classified as graphic texts.

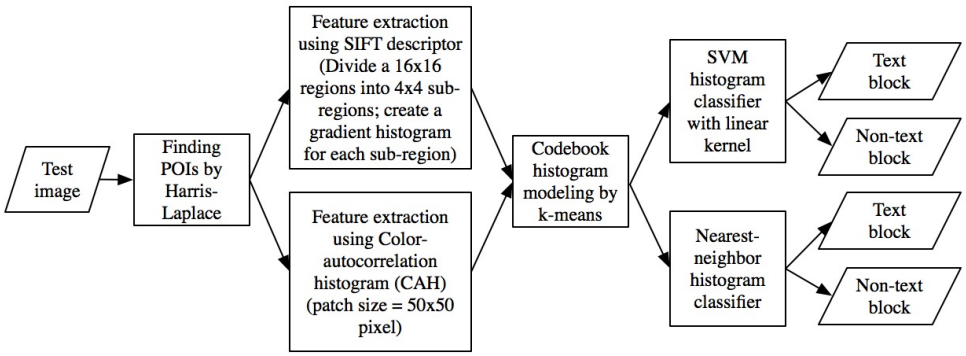


Figure 4.3: Overview of the proposed classification method.

Recently, Zhu and Zanibbi (Zhu and Zanibbi 2016) proposed a method for scene-text detection with a feature learning-based convolutional neural network called Text-Conv and cascaded classification. In the feature learning stage, they utilized and minimized some of the equations in Coates's et al. (Coates et al. 2011) algorithm. The algorithm provided the convolution masks, a number of k = 1,000, which were used for both the coarse and fine detectors stages. This stage produced many patches, and the patches were classified as text/non-text using the confidence-rated AdaBoost. The experiments regarding the Area Under Curve of Precision/Recall on ICDAR2013 (Coates et al. 2011) showed that the detection hotmap obtains 71.2%, while Coates et al. obtained 62%.

4.3 Proposed approach

For text/non-text classification, the proposed method first defines candidate text and non-text blocks from the ground truth, which will be used for generating models and test images. First, we need to locate the points of interest (POIs) in the images by, e.g., SIFT or Harris-Laplace (HL) (Section 3.2). The information of a POI (keypoint) location (x, y) is crucial to describe the image context around that detected point. The patch size of the regions of interest (ROIs) around a POI will be determined empirically (Section 4.5.1). In order to compare the capabilities of different feature-extraction techniques, we extract, at each POI, the SIFT descriptors (N_{dim} = 128) and the color autocorrelation histogram (CAH). The area used to compute

the CAH features is derived from an expanded region around the key-point position obtained from the HL detector. The image patch will be characterized by a color histogram of color space intensities, e.g., $R|G|B$ or $Y|U|V$. To suppress average lighting-condition variations, the autocorrelation function of such histograms is computed (Eq. (4.1)).

$$r(\tau) = \int_{-\infty}^{+\infty} x(t)x(t-\tau)dt \tag{4.1}$$

If the data are completely random, the autocorrelation value should be close to zero for all time lags; for instance, Figure 4.4 shows the autocorrelation plotting at time t, which should not be significantly different from the point at time $t-1$ and so on, with a peak at τ = 0. Color histograms may contain multiple intensity peaks, the position of which (i.e., the lighting) may not be important, while the presence of repetitions is significant.

lag	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X_t	0	0	0	1	2	3	4	5	4	3	2	1	0	0	0
X_{t-1}	0	0	1	2	3	4	5	4	3	2	1	0	0	0	0

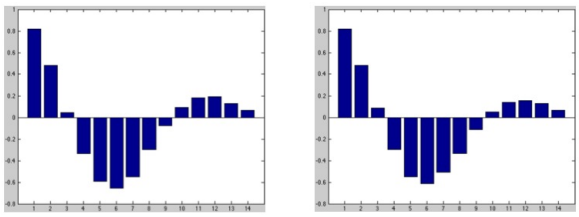


Figure 4.4: The Autocorrelation histogram of X_t and X_{t-1}.

In the proposed method, therefore, the three channels of the color space intensity histograms are shifted by autocorrelation. To reduce the dimensionality of the vector and simplify the calculation, each color channel is quantized to 128 levels and then the acf vectors of the three channels are adjoined into a vector of M_{dim} = 384. Additionally, there are various raw features derived from both the SIFT and CAH features. The BOVW technique with k-means clustering is able to reduce a huge feature-vector set into a number of k representative groups, i.e., clusters. The

Algorithm 4.1 cropBg

read total text ground truth of a training image
while $\sim eof$ **do**
compute size of each text ground truth.
compute the largest rectangle area except text ground truth of a training image.
randomly select rectangular areas from the largest rectangular area, to equal the size of the corresponding text ground truth.

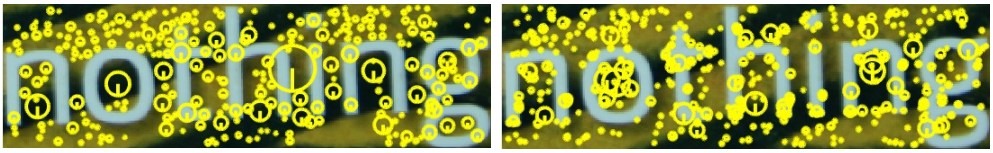
cluster centroids represent prototypical keypoints (PKP) for both FG and BG classes separately and are used as a codebook. This allows a count of the PKPs present in an image region, for both classes. Subsequently, the resulting histogram can be classified. We will test two classifiers: SVM and nearest neighbor (1NN). The diagram of the proposed classification technique is depicted in Figure 4.3.

4.3.1 Candidate text/non-text region

The method proposed in this chapter first generates candidate regions (blocks), possibly containing text (or requiring non-text) to be processed by later image analysis tools such as character extraction and recognition. To acquire text/non-text blocks, the text region can be easily derived from the ground truth, while the non-text regions need to be established automatically, with the constraint of getting sizes similar to the text regions as explained in Algorithm 4.1.

4.3.2 Localization of point of interest

A scene image usually includes illumination, texture, color and a cluttered background. Extracting a global feature from the complicated topology of the image may not provide optimal and robust results, in contrast to using local features (Farag 2014). The objective of using local features is to create precise descriptors of individual local image structures, concerning a point, edge, or corner. A local descriptor is used to indicate a small patch of pattern in an image that varies from its neighboring regions. All multiple patches are used to match an image. The matching result represents the properties of the region such as illumination, texture, and color. Good local descriptors can cover information of the image even if the image scale



(a) DoG of SIFT (267 points) (b) Harris-Laplace (HL) (450 points)

Figure 4.5: Comparison of keypoint detection described for two local feature descriptors: SIFT (ratio = 1) and Harris-Laplace.

is changed; they are also invariant to image transformations, and more robust to partial occlusion than global descriptors.

Although localization of structural features by SIFT is robust against image transformations and small geometric distortions, it sometimes provides an insufficient number of keypoints in order to obtain higher-level object attributes (Sriman and Schomaker 2015a) as shown in Figure 4.5. Figure 4.5 demonstrates the number of POIs (the yellow point) detected by SIFT and HL in the same image (Figure 4.5a shows that SIFT using the DoG algorithm provides 217 keypoints, while HL gives 421 keypoints in Figure 4.5b). The Harris-Laplace (HL) detector appears to be more effective in POI (keypoint) detection than SIFT. Therefore, this work utilizes the HL detector to detect the points of interest (POIs) in FG/BG images.

The Harris-Laplace detector, proposed by Mikolajczyk and Schmid (Mikolajczyk and Schmid 2004), has been shown to have a better performance in repeatability, localization and scale variation than other detectors. Its advantage is in providing a higher accuracy in the location and scale of the interest points with a reduced computational complexity. The Harris-Laplace detector process starts with localizing points in scale-space by the multi-scale Harris function on the image derivative P(**x**) where **x** = (x,y) using the autocorrelation matrix $\mu(\mathbf{x}, \sigma_I, \sigma_D)$. The matrix describes the gradient distribution in the local maxima of the 8-neighborhood of point **x**, which is defined by Eq. (4.2).

$$\mu(\mathbf{x}, \sigma_I, \sigma_D) = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \\ \mu_{21} & \mu_{22} \end{bmatrix} = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} P_x^2(\mathbf{x}, \sigma_D) & P_x P_y(\mathbf{x}, \sigma_D) \\ \\ P_x P_y(\mathbf{x}, \sigma_D) & P_y^2(\mathbf{x}, \sigma_D) \end{bmatrix} \tag{4.2}$$

Given the integration scale represented by σ_I where $I = 1..n$ and σ_n = $\xi^n\sigma_0$, where the scale factor between successive levels ξ = 1.4, the local scale is σ_D = $s\sigma_n$ where the constant vector s = 0.7, and the matrix $P_x(\mathbf{x}, \sigma_D)$, $P_y(\mathbf{x}, \sigma_D)$ is the derivatives computed using the Gaussian window size σ_I ($g(\sigma_I)$) in the x and y-direction at point **x**. Therefore, when the Harris measure combines the trace and the determinant of the second moment matrix, it will be changed to the function in Eq. (4.3).

$$P(\mathbf{x}) = det(\mu(\mathbf{x}, \sigma_I, \sigma_D)) - \alpha trace^2(\mu(\mathbf{x}, \sigma_I, \sigma_D)) \tag{4.3}$$

It then selects appropriate scales, which were extensively studied by Lindeberg (Lindeberg 1998). The scale selection concept is to select a characteristic scale by searching a local extremum over scales in order to reduce the set of interest points using Laplacian-of-Gaussian (LoG) in Eq. (4.4). Determined, P_{xx}, P_{yy} are the second image derivatives in horizontal and vertical directions, respectively. The characteristic scale at position **x** is defined by the scale σ_n which hits the maximum of $|LoG(\mathbf{x}, \sigma_n)|$.

$$|LoG(\mathbf{x}, \sigma_n)| = \sigma_n^2 \, |P_{xx}(\mathbf{x}, \sigma_n) + P_{yy}(\mathbf{x}, \sigma_n)| \tag{4.4}$$

Salient positions in the scale space are detected by computing a multi-scale stack of the Harris corner indicator $\{H(\mathbf{x}, \sigma_k, \gamma\sigma_k)\}_{k=1}^{n}$, where γ is a scalar. For each scale σ_k, the local maxima of the Harris indicator $H(\mathbf{x}, \sigma_k, \gamma\sigma_k)$ are computed by $\{(\hat{\mathbf{x}}, \sigma)\} = argmaxlocal_{\mathbf{x}} H(\mathbf{x}, \sigma_k, \gamma\sigma)$. Then, select points at which the normalized LoG is maximal across scales and the maximum is above a threshold ($\hat{\sigma} = argmax_{\sigma}$ $|LoG(\hat{\mathbf{x}}, \gamma\sigma)|$). The spatial location $\hat{\mathbf{x}}$ which do not hit a scale maximum, is given up.

4.3.3 Feature extraction

The POIs or keypoints obtained from the previous process will be used to compute dedicated features. Around each keypoint location (Figure 4.6a and Figure 4.6c), two types of feature are computed, i.e., the SIFT descriptor and a color autocorrelation histogram (CAH). Suppose there is a candidate FG image with HL-detected

keypoints as POIs in Figure 4.6. At each extracted candidate keypoint (Figure 4.6a), a SIFT descriptor will be computed (Figure 4.6b). At the same time, the expanded region around the POI candidate will be cut out of the image, i.e., be 'cropped' with a width equal to the length in Figure 4.6c. The small, cropped region of M×M pixels will be called ROI (region of interest) in the sequel. Then the color autocorrelation histogram feature (CAH) will be computed from this ROI (Figure 4.6d). As a result, the feature description addresses both structural shape aspects (SIFT) and color information (CAH).

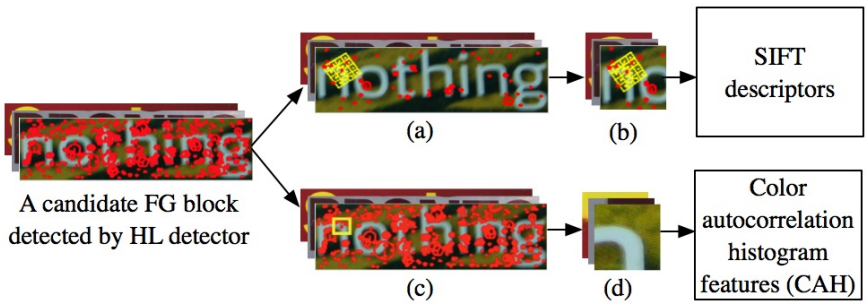


Figure 4.6: Extracting features at each keypoint using SIFT and CAH.

Scale invariant feature transform (SIFT)

SIFT is an efficient local feature descriptor proposed by Lowe (Lowe 2004), which has been shown to be useful in extracting dominant image features regardless of translation, scaling and projective transformations on the salient visual element. SIFT includes four major stages: 1) scale-space extrema detection, 2) keypoint localization, 3) orientation assignment, and 4) computing the keypoint descriptors. The scale space of an image is defined by $L(x, y, \sigma)$ obtained from the convolution of an image $I(x, y)$ with the Gaussian filter $G(x, y, \sigma)$ with a small σ value (Eq. (4.5)).

$$\begin{aligned} L(x, y, \sigma) &= G(x, y, \sigma) * I(x, y) \\ G(x, y, \sigma) &= \tfrac{1}{2\pi\sigma^2} e^{-((x^2+y^2)/2\sigma^2)} \end{aligned} \tag{4.5}$$

Extracting the SIFT feature requires the difference between two Gaussian-convo-

lved images to be computed, at different scales, by using Different-of-Gaussian (DoG), which is separated by a constant multiplicative factor $k = \sqrt{2}$ in Eq. (4.6). In order to obtain accurate keypoint localization, the extrema (min and max) need to be computed by comparing a pixel to its neighbors in the filtered image.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{4.6}$$

Therefore, we extract local features of various resolutions and scales at the POIs, which are detected by the Harris-Laplace keypoint detector (represented by the red circles in Figure 4.7a). Figure 4.7b shows an example of POI features extracted by SIFT features, which consist of a coordinate (x, y), scale, and orientation in adjacent 4×4-pixel sub regions of a 16×16-pixel square. The descriptors are computed using the gradient magnitude and orientation around the keypoints, i.e. eight values per sub region. Then the 16×8 = 128-dimensional feature descriptor of this keypoint is complete, as demonstrated in Figure 4.7c.

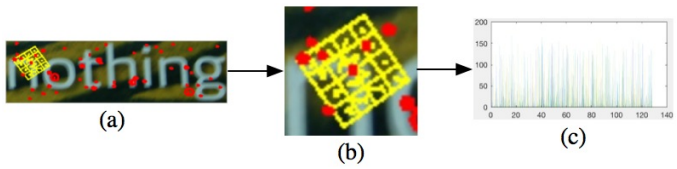


Figure 4.7: An example of generating features at each keypoint using SIFT descriptors.

Color intensity histogram by time series analysis

The text and non-text blocks appearing in nature scenes are usually decomposed in different color channels. Color is a determinant of the image that plays a vital role in image-retrieval systems. Color or hue is often applied in scene-image processing because color can be highly informative in object classification of both natural objects (e.g., fruits, flowers) and manmade artefacts such as colorful advertisement texts and street signs. Furthermore, color also facilitates the distinction in homogeneous image regions, such as blue sky, the yellow of banana, or the green of leaves. In the

image-processing process, a system needs to be able to associate the measured color intensities with physical properties. This is not easy, due to the interaction between the color properties of the incident light and the light that is reflected by an object (metamerism). There are many color spaces that have been used in color analysis (Torp et al. 1994). In order to choose the most accurate color space for the purpose of image classification in this chapter, a set of the well-known color spaces RGB, $C_1C_2C_3$, $L_1L_2L_3$, $O_1O_2O_3$, HSI, HSV, $I_1I_2I_3$, YUV and YIQ is analyzed.

1. RGB color images are generally contained in the format of the RGB color space Eq. (4.7), which is blended from the three primary colors red, green and blue. Different proportions of the three colors can make more colors. These colors are used for light display systems such as television, computers, cameras, and projectors.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} R/(R+G+B) \\ G/(R+G+B) \\ B/(R+G+B) \end{bmatrix} \tag{4.7}$$

2. $C_1C_2C_3$ is a normalized RGB, which has invariant characteristics. The $C_1C_2C_3$ color space is appropriated for photometric color invariants for matte, dull surfaces (Gevers and Smeulders 1999). It is independent of the changes in orientation, illumination direction, and illumination intensity; the color is determined by Eq. (4.8).

$$\begin{bmatrix} C1 \\ C2 \\ C3 \end{bmatrix} = \begin{bmatrix} tan^{-1} \times (R/max\{G,B\}) \\ tan^{-1} \times (G/max\{R,B\}) \\ tan^{-1} \times (B/max\{R,G\}) \end{bmatrix} \tag{4.8}$$

3. $L_1L_2L_3$ proposed by Gevers and Smeulders (Gevers and Smeulders 1999), was presented to determine the direction of a triangular plane in the RGB space, or it represents normalized square color differences as described in Eq. (4.9).

$$\begin{bmatrix} L1 \\ L2 \\ L3 \end{bmatrix} = \begin{bmatrix} (R-G)^2/((R-G)^2+(R-B)^2+(G-B)^2) \\ (R-B)^2/((R-G)^2+(R-B)^2+(G-B)^2) \\ (G-B)^2/((R-G)^2+(R-B)^2+(G-B)^2) \end{bmatrix} \tag{4.9}$$

4. $O_1O_2O_3$, the opponent color space, is color separated into three channels that are channel O_1, O_2, O_3. Channel O_1 and O_2 represent the chromatic information, while O_3 represents the intensity information in Eq. (4.10).

$$\begin{bmatrix} O1 \\ O2 \\ O3 \end{bmatrix} = \begin{bmatrix} (R-G)/\sqrt{2} \\ (R+G-(2\times B))/\sqrt{6} \\ (R+G+B)/\sqrt{3} \end{bmatrix} \tag{4.10}$$

5. HSI, the hue-saturation-intensity color model, is human intuition and commonly used in image processing. The HSI in Eq. (4.11) represents three elements: hue (H) describes its color in the range [0°,360°], saturation (S) reflects the color purity, and intensity (I) is black and white. HSI is generated via the nonlinear transformation of the RGB color space, in which the intensity and saturation of manipulated pixels is limited in the range of [0,1].

$$\begin{bmatrix} H \\ S \\ I \end{bmatrix} = \begin{bmatrix} arctan(\sqrt{3}(G-B)/(2R-G-B)) \\ 1-min(R,G,B)/I \\ (R+G+B)/3 \end{bmatrix} \tag{4.11}$$

6. HSV is a color model comprising of hue, saturation, and value, that describes color in terms of its shade and brightness. Hue is the value of the primary colors (red, green and blue). Value is the brightness of the colors, which can be measured by the intensity of the brightness of each color. Therefore, HSV in Eq. (4.12) can expediently make the brightness channel more vivid than other color models.

$$\begin{bmatrix} H \\ S \\ V \end{bmatrix} = \begin{bmatrix} arctan(\sqrt{3}(G-B)/(2R-G-B)) \\ 1-3\times min(R,G,B)/(R+G+B) \\ max(R,G,B) \end{bmatrix} \tag{4.12}$$

7. $I_1I_2I_3$ color space is obtained through the deceleration of the RGB color components using the dynamic K.L. transformation by Ohta et al. (Ohta et al. 1980). The

three orthogonal color spaces are according to Eq. (4.13).

$$\begin{bmatrix} I1 \\ I2 \\ I3 \end{bmatrix} = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.5 & 0.0 & -0.5 \\ -0.25 & 0.5 & -0.25 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.13}$$

8. The YUV color space in Eq. (4.14) is used to color video standards. It most added on phase alternating line (PAL) and sequential color with memory (SECAM) television. It consists of the luminance (Y) component, which determines the brightness of the color, while the two components (U and V) represent the color itself (the chrominance).

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ -0.15 & -0.29 & 0.44 \\ 0.6 & -0.51 & -0.1 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.14}$$

9. YIQ is implemented for TV broadcasting, which the primary objective is working with black and white television. The composition of YIQ is represented by Y, I and Q, which is an encoded color signal of the image. The Y represents the luminance information, which is the signal used for black and white TVs. The chrominance information is contained in the components I and Q. The procedure of YIQ is demonstrated in Eq. (4.15).

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ 0.6 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.15}$$

In order to obtain a usable degree of independence from lighting conditions, we propose to use an autocorrelation on the histograms of the color channel intensities. The autocorrelation-based CAH feature should represent color distribution in a manner which is not directly determined by the general intensity level, while retaining relative color information. There are different practical methods for computing

an autocorrelation function. Some of them operate via the time domain, whereas others are based on the inverse Fourier transform. Additional implementation details involve normalization and handling of circularity and/or boundary effects. We used three methods: the autocorrelation function (ACF) in Figure 4.8, the inverse fast Fourier transform (IFFT) in Figure 4.9, and the cross-correlation (XCORR) in Figure 4.10 (Matlab, (*xcorr Cross-correlation* 2017)), in order to find the optimal method for discriminating between foreground and background regions. Each patch region is separated into three channels of a given color space intensity histogram, for example, $R|G|B$ color space, or $H|S|V$, etc. The resulting histogram is then converted by one of these three kinds of autocorrelation. In order to reduce computational complexity, the histogram has reduced dimensions (128 levels). Finally, the autocorrelation vectors of the three channels are adjoined into an M_{dim} = 384 vector to represent the complete CAH feature.

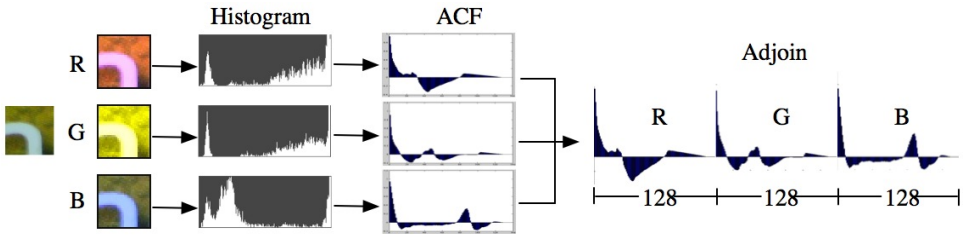


Figure 4.8: The process of feature extraction on an image patch based on the color intensity histogram with the autocorrelation function (ACF).

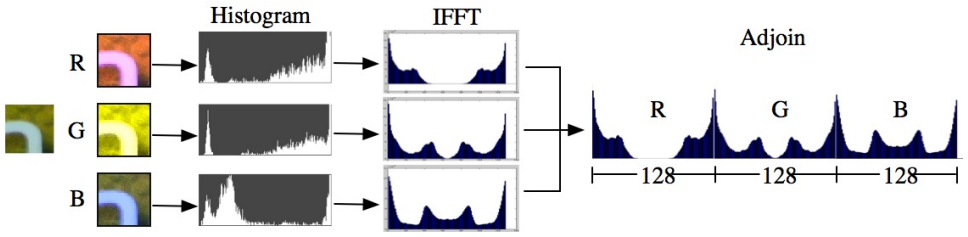


Figure 4.9: The process of feature extraction on an image patch based on the color intensity histogram with the Inverse fast Fourier transform (IFFT).

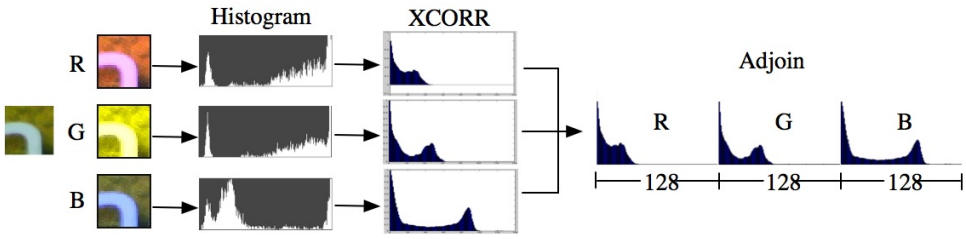


Figure 4.10: The process of feature extraction on an image patch based on the color intensity histogram with the cross-correlation (XCORR).

4.4 Codebook histogram modeling

The bag of features model has been widely used in computer vision and video analysis, e.g., for separating objects from cluttered backgrounds, for face and character recognition, and for image classification. In this chapter, the bag of visual words (BOVW) is deployed to create a combined codebook histogram of color autocorrelation histogram (CAH) features and SIFT features. When all the training images have been processed, yielding the CAH features and SIFT descriptors, the codebook generation is performed through clustering as depicted in Figure 4.11.

In order to obtain an optimum codebook for CAH, there are design decisions as regards the number of elements k of the codebook histogram and the use of distance function. The mobile device platform poses restrictions on memory use and computing efficiency. In addition, it is essential to determine the optimal local patch size to determine the level of detail to describe a cropped region. Various color spaces need to be evaluated for the CAH feature. Furthermore, a proper normalization method is necessary for calculating data in an adjoined vector with values in a comparable range. The relative performance of CAH and SIFT descriptors needs to be evaluated. Consequently, in order to choose the optimal parameter values, all the conditions in Section 4.5.1 need to be compared in terms of the performance of text versus non-text classification. We will use statistical testing (ANOVA) for the selection process.

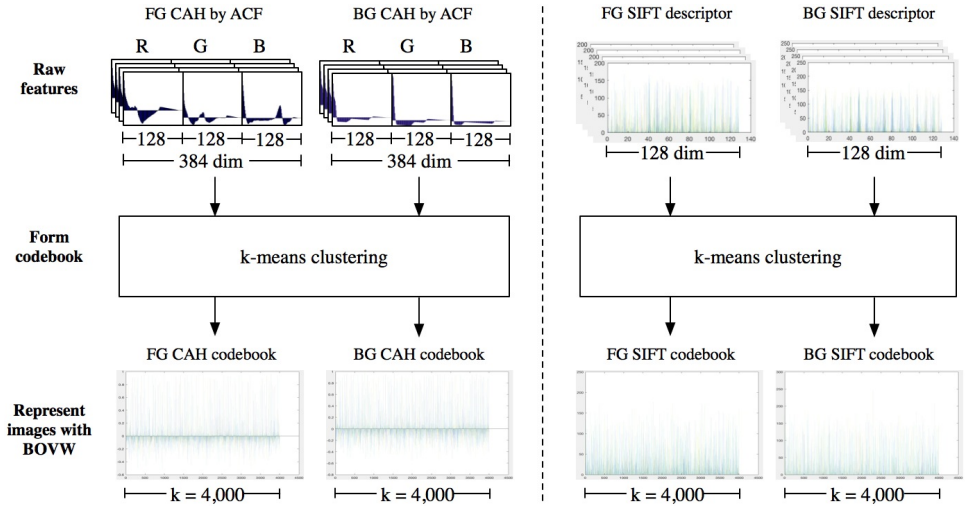


Figure 4.11: The process of FG/BG codebook histogram creation using color autocorrelation histogram (CAH) features (left-hand side). The FG and BG codebooks of SIFT features are realized by using k-means with k = 4,000 in order to be comparable with CAH (right-hand side).

4.4.1 K-means clustering

The concept of clustering is to find similar characteristics between data without defined data classes or when the exact number of groups is unknown, in order to group similar objects into clusters. The process is called an 'unsupervised learning algorithm'. K-means by Lloyd's algorithm (Lloyd 2006) is an unsupervised learning algorithm that is used for solving clustering problems in the cluster analysis. The concept of k-means is developing a lexicographic classification for a huge sample of data. Its purpose is to partition an N-dimensional population into the basis sample number of clusters (assume k clusters).

The process of k-means is as follows:

1. Set s as a dimension of feature descriptors d_1, d_2, ..., d_s, where $d_{1..s}$ are members of a set of keypoints (kp).

2. Suppose that we have n keypoint (kp) vectors kp_1, kp_2, ..., kp_n all from the same class, and let k be the number of clusters to be grouped, which $k < n$.

3. Define the initial clusters by using random m keypoints as a partition.

4. Compute the cluster mean of each individual group.

5. Calculate the distance between cluster members and all cluster means (centroids) to find the nearest centroid of each member.

6. Iteratively relocation members to an appropriate cluster based on calculated distance until clusters are explicitly separated.

Therefore, we run k-means several times with different numbers of desired representative vectors (k) and different sets of initial cluster centers. We select the final clustering giving the lowest empirical risk in categorization.

4.5 Datasets and training process

We assess the performance of classification on three benchmark datasets including Western and Asian scripts. Data characteristics are urban/natural scenes with embedded text image information, in general with highly variable color, contrast, and gradient information. Overall images often have a complex background with similar colors and gradients as the text scale, font variations, and orientation. Also, shadow and lighting conditions are variable. Text styles (fonts) are multi-color and highly variable. The image regions containing text are provided as ground truth labels. The three proposed word scene classification methods are described as follows.

The first dataset is the Chars74K image dataset (in Figure 4.12a) (de Campos et al. 2009), which uses Kannada script that is similar to Thai script. The Kannada contains 390 text images with a minimum size of 37×60 pixels and maximum of 300×27 pixels, and the average height of the characters is 93.77 pixels. The dataset will be used to tune the control parameters and feature method.

The second is the ICDAR2015 (Karatzas et al. 2015), a well-known dataset, containing 1,943 Western word images (in Figure 4.12b). It comprises an image size of 16×67 pixels for minimum and maximum of 300×42 pixels, and the average height of the characters is 87.85 pixels.

The third dataset is the Thai scene image dataset (TSIB) (Sriman and Schomaker 2015b) captured by a smartphone (in Figure 4.12c). All the words appearing in images are manually annotated. This dataset contains 10,400 word images. The minimum and maximum sizes are 8×11 pixels and 1,485×415 pixels, respectively. The

average height of the characters is 84.45 pixels. A random selection of 5,200 text images will be made to conduct the experiments.

(a) Chars74K dataset (b) ICDAR2015 dataset (c) TSIB dataset

Figure 4.12: Example text (FG) and non-text (BG) images from (a) the Chars74K, (b) ICDAR2015, and (c) TSIB datasets.

4.5.1 Training process

When computing color feature images in different color spaces, there are various qualities and different discriminating capabilities of the images since the color is mixed up with different elements. Each color element or space, e.g., the RGB color model, is an additive model, which comprises the elements red, green, and blue (that are used to select the optimum criteria). At each step of the patched regions, the novel color feature using autocorrelation on the single-channel is calculated for the pixels in that region and used to compute the intensity histograms. During the stage of exploring the control parameters and feature methods in order to study the impact of different conditions for CAH based on RGB color, we explore the optimal parameter values for: i) patch sizes, ii) autocorrelation types, iii) distance types, iv) normalization types, and v) codebook sizes which requires a computationally intensive grid search. All the experiments were tested on the Peregine cluster, which has 24 cores @ 2.5 GHz (two Intel Xeon E5 2680v3 CPUs). The details of the conditions are defined as follows.

1. patch sizes: 20×20, 30×30, 40×40, 50×50, and 60×60 pixels,

2. autocorrelations: XCORR, IFFT, and ACF,

3. distances: cosine, correlation, Euclidean, city block, Spearman, and Chebychev distances,

4. normalisations: i) (x-xmin)/(xmax-xmin),
   ii) (x-mean(x(:)))/std(x(:)),
   iii) (|x|-xmin)/(xmax-xmin),
5. k in k-means clustering: 1,000, 2,000, 3,000, 4,000, 5,000, 6,000, and 7,000.

Experiment 1: This experiment was conducted to obtain the optimum control parameters. The first stage, we use the training set of Kannada script in the Chars74K dataset, which was divided into 10 folds. The experiment starts with all the training images being extracted to get CAH feature descriptors, which are collected into a file and labeled separately with FG and BG. Then 1,000 FG and BG descriptors are selected randomly and matched to the FG and BG codebooks. The matching results show the probability of correct matching as explained in Figure 4.13. A selection of the 105 matching results were analyzed by the ANOVA statistics. The analysis was computed 5 times (525 in total). We selected the best ANOVA result to get effective criteria that perform proper classification to be used in the next step.

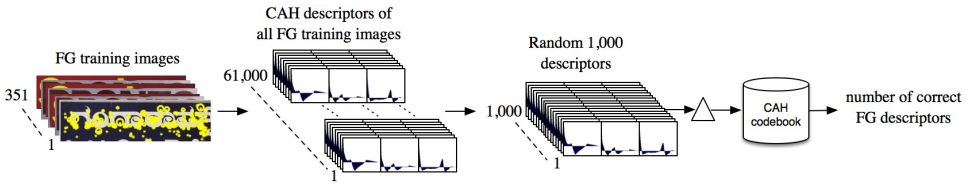


Figure 4.13: The process of computing to find the optimum criteria.

Computing the difference of the training set; the result is illustrated in Table 4.1. It shows that the mean difference is not significant: 0.243 ($p \gg 0.05$), and therefore the data in the groups have no distribution difference.

Table 4.1: ANOVA statistic for the average results of the training set of 5-fold.

Fold	N	Mean	SD	Std. Err	95% CI		Min	Max
					Low	Up		
1	105	97.80	2.43	0.24	97.33	98.28	85.40	99.95
2	105	97.90	2.21	0.22	97.47	98.33	85.30	99.80
3	105	98.07	2.20	0.21	97.65	98.50	86.50	99.95
4	105	97.96	2.35	0.23	97.51	98.42	86.90	99.90
5	105	98.44	1.51	0.15	98.15	98.74	89.10	99.95
Total	525	98.04	2.17	0.09	97.85	98.22	85.30	99.95

The area of the region of interests (ROIs) or patch size (derived from the exten-

sion of the POI by HL) was included for testing in the experiments. From Table 4.2, the mean difference is tested and it is found that the difference is significant at $p \ll 0.000001$, and the window size affects the accuracy, the wider window, and the higher accuracy. This is because the average height of the characters in the three datasets is 88.69. Hence, the window size must be large enough, e.g., the patch size should be higher than half the height of the character. Therefore, the patch size 50×50 is selected.

Table 4.2: Evaluation of accuracy over patch sizes.

Patch Sizes	N	Mean	SD	Std. Err	95% CI Low	95% CI Up	Min	Max
20×20	105	96.49	3.12	0.30	95.89	97.09	85.30	99.00
30×30	105	97.70	2.03	0.20	97.31	98.09	89.75	99.65
40×40	105	98.48	1.36	0.13	98.22	98.74	91.75	99.80
50×50	105	**98.61**	1.55	0.15	98.32	98.91	91.90	99.90
60×60	105	98.90	1.34	0.13	98.64	99.16	92.00	99.95
Total	525	98.04	2.17	0.09	97.85	98.22	85.30	99.95

As mentioned earlier, selecting the type of autocorrelation is a crucial step in getting the promising result; therefore, a comparison of the results of the autocorrelation techniques: IFFT, XCORR, and ACF are demonstrated in Table 4.3. The mean difference is significant at $p \ll 0.000001$. Thus, the ACF algorithm affects the accuracy.

Table 4.3: ANOVA statistic for training set of three types of CAH.

Autocorrelation	N	Mean	SD	Std. Err	95% CI Low	95% CI Up	Min	Max
IFFT	175	98.01	1.30	0.10	97.82	98.21	93.65	99.70
XCORR	175	97.53	3.00	0.23	97.08	97.97	85.30	99.90
ACF	175	**98.57**	1.71	0.13	98.31	98.83	89.10	99.95
Total	525	98.04	2.17	0.09	97.85	98.22	85.30	99.95

There are many distance metrics that can be used in creating a model. In order to delineate the relation between average accuracy and the distance metrics, ANOVA computes the mean difference and shows that it is significant at $p \ll 0.000001$, and that the city block distance algorithm relates to the accuracy (Table 4.4).

The normalization method is used to handle values in the same range. The three normalization algorithms: i (*), ii (**), and iii (***) (in Section 4.5.1) are taken into account. From the empirical experiments, the mean difference is significant at $p \ll 0.000001$; thus, algorithm number ii matters for the accuracy (Table 4.5).

Table 4.4: Evaluation of accuracy over distance functions for 1NN.

Distances	N	Mean	SD	Std. Err	95% CI Low	95% CI Up	Min	Max
cosine	275	98.37	1.22	0.07	98.22	98.51	93.65	99.95
correlation	132	98.36	1.80	0.16	98.05	98.67	90.55	99.95
Euclidean	59	98.82	0.85	0.11	98.59	99.04	94.85	99.90
city block	16	**98.84**	1.39	0.35	98.10	99.58	94.50	99.95
Spearman	43	93.56	3.80	0.58	92.39	94.73	85.30	99.75
Total	525	98.04	2.17	0.09	97.85	98.22	85.30	99.95

Table 4.5: Evaluation of accuracy over normalization methods.

Normalizations	N	Mean	SD	Std. Err	95% CI Low	95% CI Up	Min	Max
i (*)	489	98.24	1.73	0.08	98.09	98.40	85.40	99.95
ii (**)	16	**98.83**	0.79	0.20	98.41	99.25	97.05	99.80
iii (***)	20	92.36	4.08	0.91	90.45	94.27	85.30	99.80
Total	525	98.04	2.17	0.09	97.85	98.22	85.30	99.95

* i) (x-xmin)/(xmax-xmin)
** ii) (x-mean(x(:)))/std(x(:))
*** iii) ($|x|$-xmin)/(xmax-xmin)

To verify the codebook size of each FG and BG, we varied the size of codebook between 1,000 - 7,000 and increased the step of the codebook by 1,000 as shown in Table 4.6. The mean difference is significant at $p \ll 0.000001$, which means that the size of codebook affects the accuracy. A higher size of codebook gives a higher accuracy. However, to make the processing time more applicable, but still productive, we chose 4,000 for generating the codebook size of both FG and BG.

Table 4.6: Evaluation of codebook size for SIFT and CAH descriptors.

Clusters	N	Mean	SD	Std.	95% CI Low	95% CI Up	Min	Max
1,000	75	95.35	3.09	0.36	94.64	96.06	85.30	98.80
2,000	75	97.06	2.04	0.24	96.59	97.53	89.95	99.25
3,000	75	98.21	1.74	0.20	97.80	98.61	86.90	99.70
4,000	75	**98.53**	1.41	0.16	98.21	98.85	90.55	99.75
5,000	75	98.77	1.31	0.15	98.47	99.08	90.55	99.95
6,000	75	99.10	0.92	0.11	98.89	99.31	93.15	99.95
7,000	75	99.24	0.58	0.07	99.10	99.37	97.45	99.90
Total	525	98.04	2.17	0.09	97.85	98.22	85.30	99.95

4.6 Classifier design

Consequently, the optimal parameters tested by ANOVA in Section 5.1 are used to establish a color autocorrelation histogram model (CAH). The CAH model is used

Algorithm 4.2 NN voting (v, pkp)

correctFg = 0; correctBg = 0;
$cb = \{pkp_1, pkp_2, .., pkp_m\}$;
$cf = \{v_1, v_2, .., v_n\}$;
[ncb, ncf] = normalize(cb, cf);
$sepCB = size(cb, 1)/2$;
$[D, I] = pdist2(ncb, ncf, distance,' Smallest', 1)$;
$firstHalf = sum(I \leq sepCB)$;
$secondHalf = sum(I > sepCB)$;
if $firstHalf > secondHalf$ **then**
  if $GT ==' fg'$ **then**
    $correctFg = correctFg + 1$;
    $decision =' fg'$
else
  if $GT ==' bg'$ **then**
    correctBg = correctBg + 1;
    $decision =' bg'$
return $decision$

to determine the validity of the FG/BG classification. In this task, the model is constructed from two classifier techniques: nearest neighbor (1NN) and support vector machine (SVM), for comparison purposes.

4.6.1 Nearest neighbor voting

The nearest neighbor (1NN) is a simple algorithm that can be used for classification even on the basis of a few examples. The advantage of this approach is that it is not complicated. However, the computation time can vary depending on the size of the reference dataset. The goal is to compare the similarity between unknown objects and neighboring candidates, by calculating the distance in some feature spaces. The closest neighbor indicates the class of a query object.

Foreground and background class, shown in Algorithm 4.2, are defined and denoted as 'fg' and 'bg'. Each class has an associated codebook, which is represented by prototypical keypoints (PKPs). Given an image (I), PKPs are detected. Each PKP is classified into either fg or bg. The final decision is made based on the majority type of the PKPs voting scheme. However, since this voting scheme appears to be highly simplistic, it was assumed that a richer description of images should be used

to take the decision. In the next section, we will introduce a codebook histogram method, using SVM for classification.

4.6.2 Support vector machine (SVM)

In this approach, image regions are characterized by a distribution (histogram) of prototypical keypoints; again, there are two codebooks, one for FG and one for BG. Instead of using a sample vote technique, image descriptions (i.e., codebook histogram) will be compared vectorially, using a support vector machine. Support vector machines (SVMs) (Cortes and Vapnik 1995) in machine learning is a supervised learning model, which is able to handle data analysis and classification problems. The algorithm is based on the principle of finding the coefficients of the equation to create a dividing line of data that is sent to the training process by focusing on the best dividing line of data. SVM with the linear kernel is indeed one of the simplest classifiers with a lower risk of overfitting than non-linear kernels such as polynomial or RBF. Since we perform a grid search over many parameters, using the linear SVM is applicable for the task. SVM can be used to identify patterns or groups of data; the data are divided into two sides by the hyperplane. Initially, the hyperplane is usually formed in the linear model also used in this research, and the appropriated linear model is selected by SVM with the maximum distance between the two classes (the so-called functional margin). The margin means the maximal width of the slab parallel to the hyperplane that has no interior data points.

Given a set of feature X = $(x_1, y_1), ..., (x_n, y_n)$ when $x \in R^m$ where n is a number of sample data, m is the dimension of input data, x is the feature vector, and y is a group of data which includes $\{-1, 1\}$. Then set the linear equation to establish a straight line on the hyperplane to separate the data into two groups which are represented by the value of y. The data that determines the slope and plane formed on the hyperplane is derived from the pair of (w, b), where w is the slope value and b is the constant value (Y-axis value). Defining the equation identifies which part of the group is located on the hyperplane as shown in Eq. (4.16) and Eq. (4.17). Applying all the conditional formulas to the geometric analysis by considering where the data

are grouped in accordance with SVM is depicted in Eq. (4.18).

$$w^T x + b \geq y, \; where \; y = 1 \tag{4.16}$$

$$w^T x + b \leq y, \; where \; y = -1 \tag{4.17}$$

$$y(w^T x + b) - 1 \geq 0 \tag{4.18}$$

However, sometimes it is not possible to distinguish all the data correctly. Therefore, reducing these errors can be a crucial step in improving the classification performance. Thus, it is necessary to define a variable to accept the error value to SVM. The reconstruction of SVM is explained by $\phi(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$. The formula describes the vector of the weight values of w by trying to reduce the value in the first term of the equation to the smallest value. C is a variable that can be customized to adjust the desired error value as low as possible. The measurement of errors that deviate from the correct position is defined by ξ_i or slack variable.

To train an SVM model, there are several hyperparameters to be considered. For instance, the different values for parameter C will yield different performance results. Using the fitcsvm() procedure, we determined the following values for C. This research used C = 0.3161 for the color model and C = 0.0010 for the SIFT model of the TSIB dataset. For the Chars74K dataset with the Kannada script, C = 981.45 was utilized for the color model and C = 332.51 for the SIFT model, and C = 0.0010 was utilized for the color model and C = 0.0011 for the SIFT model of the ICADAR2015 dataset.

4.7 Experimental results

In this section, we demonstrate the result on the three benchmark datasets. We designed the experiments in order to investigate the influence of various color spaces,

color space against SIFT, as well as classifier methods on classification performance. Furthermore, we take advantage of the SIFT feature to remind the object description and color autocorrelation histogram to compute the color distribution of an object, and to create adjoined features and test for classifying text and non-text blocks. To evaluate the performance of the proposed method, we use the precision, recall and f-measure. Here precision: $p = TP/(TP + FP)$, Recall: $r = TP/(TP + FN)$ where TP is the set of true positive classification while FP is Type I error and Type II error for FN. Therefore, the definition of f-measure is $(f = 2/(1/p + 1/r))$.

Experiment 2: The image-patch regions of interest are represented by a histogram of color autocorrelation (CAH features). It is interesting to observe the classification performance of CAH features. There are nine color spaces using the selection criteria from Section 4.5.1 for creating CAH feature, to be assessed. From the selective measurements it can be concluded that the optimal patch region size of 50×50 pixels is optimal for describing the foreground (FG) or text elements. Since the average height of a character in the three datasets is 88.69 pixels, hence the found window size is apparently large enough. For example, as a rule of thumb, the patch size should be higher than half the height of the character. Smaller patch sizes may not indicate clearly whether a region of interest concerns FG text or BG scene content.

Another important factor is autocorrelation, which was computed using three different autocorrelation algorithms: IFFT, XCORR, and ACF. Each color channel is quantized with a similar level to 128 dimensions and is adjoined into 384 dimensions to compare the classification results. The ACF is normalized by mean and variance, or in other words has 'autocovariance function'. The ACF pattern has a large spike at lag 1 followed by a decreasing wave after a few lags that alternate between positive and negative correlations. Then, a multiple-dimension IFFT is a symmetric pattern or a complex conjugate; it has considerable computational advantages over the convex optimization method (Zhang and Su 2016). On the other hand, XCORR uses cross-correlation which computes the normalized autocorrelation sequence to lag 255 by 'coeff' followed by shifting an odd position to lag 128. Therefore, the sequence is so that the autocorrelations at zero lag equal to 1. We qualitatively compare the efficiency of three autocorrelations in Section 4.5.1 (Table

4.3). The result shows that the ACF provides better classification results than IFFT or XCORR concerning both mean and maximum values by ANOVA. On the basis of this result the ACF method seems to be the most appropriate to compute a color autocorrelation histogram feature.

To demonstrate the effectiveness of the distance functions in order to create the 1NN classifier, six distance types are examined. In ascending order of the mean, the Spearman gives the lowest mean at 93.56%, followed by the correlation, cosine, and Euclidean at 98.36%, 98.37%, and 98.82%, respectively. The best distance measure is the city block (Manhattan) function with a mean score of 98.84%. Therefore, city block is selected for applying 1NN matching. In addition, the normalization is also advantageous to handle values in a comparable range: the best method is ii(std), yielding a mean performance of 98.83%.

The final factor is the size of codebook, which can greatly affect the performance. We tested the codebook size 1,000 to 7,000 for the range of a thousand. A higher codebook size gives a higher accuracy, as depicted in Section 4.5.1 (Table 4.6). However, if the codebook size is too large, several related region features will be matched to different visual words and take a lot of time. In contrast, if the codebook size is very small, many unrelated regions will be matched to the same visual words. Therefore, to make the processing time-saving but still effective, we choose 4,000 for creating the codebook size of FG and BG.

The experimental results of FG and BG classification with 1NN classifier of three datasets on the criteria in Section 4.5.1 can be demonstrated in Figure 4.14. From the figure we observe that there are three color spaces; HSI, $I_1I_2I_3$, and YIQ provide 90% high accuracy for Chars74K. On the other hand, the ICDAR2015 needs four colors: $C_1C_2C_3$, HSI, RGB, and YUV, which gives 73% accuracy. Finally, RGB and YUV color are appropriated for the TSIB dataset with performance accuracy at 88%.

Experiment 3: Detecting POIs by HL is more useful for POIs (keypoint) detection than SIFT (Figure 4.5) because a large number of POIs would obtain higher-level object attributes, e.g., descriptions and color. Therefore, this work utilizes the HL detector to detect the points of interest (POIs) of FG/BG images. Each POI is

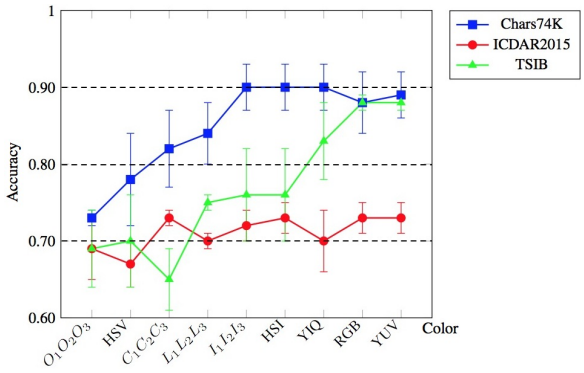


Figure 4.14: The accuracy of FG and BG classification with a 1NN classifier using the different types of color spaces, for the three datasets. Color scheme RGB and YUV appear to perform well on sets Chars74K and TSIB, while the performance on ICDAR2015 is low, for all the color schemes.

an extracted feature in 2 types: SIFT and CAH. Extracting the feature descriptor using SIFT gives 128 dimensions at each POI; on the contrary, extracting features in different color spaces after applying ACF at each ROI has different characteristics but similar dimensions to 384. At this stage of comparing text/non-text classification, we want to test the basic features first, so that they, for example CNN-based methods, can be contrasted with this in later research.

The accuracy of CAH and SIFT features is compared using two different model classifier methods: the ordinary nearest neighbor (1NN) and a proven method (SVM) as demonstrated in Figure 4.15. Figure 4.15 describes the results of FG and BG classification with the 1NN classifier compared to the SVM classifier using the different kinds of color space for the three datasets. Since the SIFT model is created only once, it obtains similar results in each graph. The performance of SIFT using the 1NN classifier is better than the color features (CAH) using the same classifier at approximately 90% for the Chars74K data set. However, the performance of SIFT by the SVM classifier presents lower accuracy than 1NN at approximately 86% for similar data sets. Surprisingly, when considering a TSIB data set, CAH beats SIFT by two color spaces that are RGB and YUV (around 88%). Meanwhile, the performance of CAH using 1NN and SVM is still lower on average for FG/BG classification on the ICDAR2015 data set than for the other two data sets.

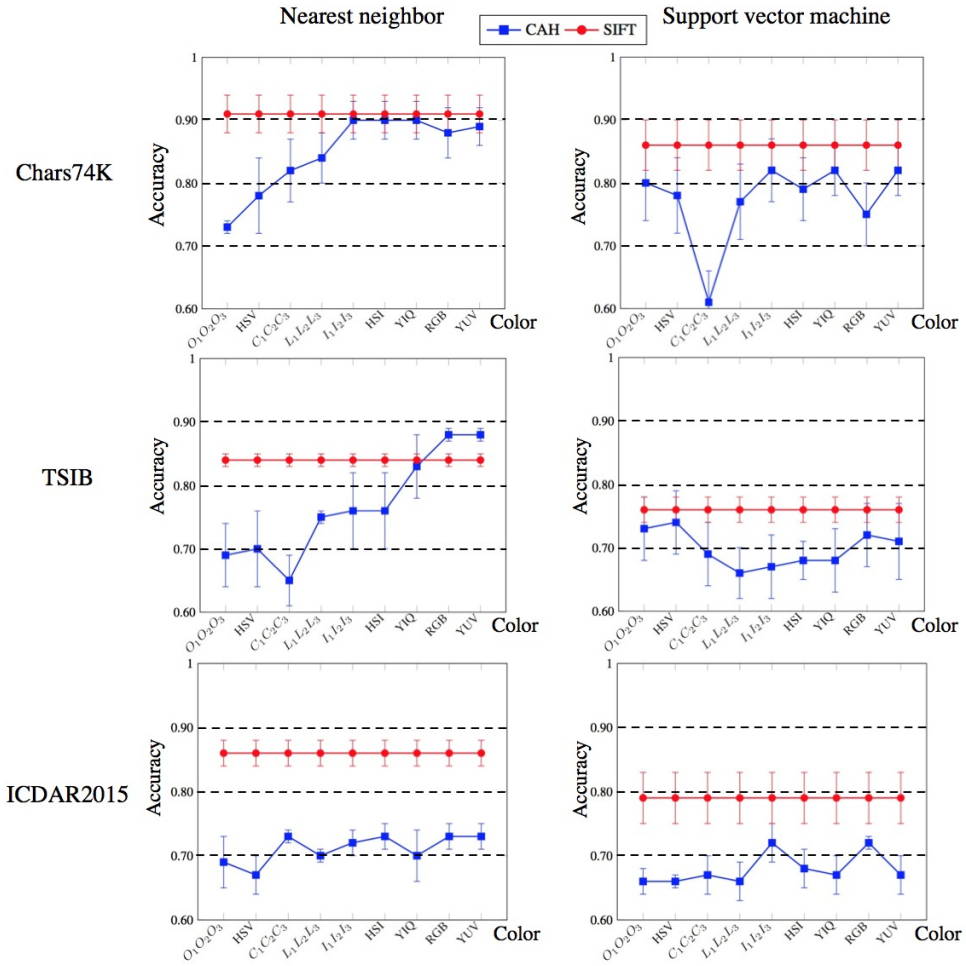


Figure 4.15: The results of FG and BG classification with the 1NN classifier compared to the SVM classifier using the different types of color spaces for the three datasets. The performance of SIFT is better than the color feature (CAH).

Table 4.7 and Table 4.8 show the evaluation via precision, recall, and f-measure for the classification results of the CAH and SIFT features using 1NN classifier and SVM classifier, respectively, on the three benchmark datasets. The performance of the CAH feature is comparable to state-of-the-art features such as SIFT. Although SIFT yields better results than the CAH feature for both 1NN and SVM classifiers,

we also observe that the f-measure on the CAH feature of TSIB (a colorful data set) in Table 4.7 and Table 4.8 gives better results compared to SIFT. This gives an indication for the application domain of the CAH feature for classifying text and non-text blocks in images.

Table 4.7: The criteria performances of CAH and SIFT features using 1NN classifier for Chars74K, TSIB, and ICDAR2015 datasets.

Color space	Chars74K			TSIB			ICDAR2015		
	p	r	f	p	r	f	p	r	f
$O_1O_2O_3$	0.51	0.90	0.65	0.90	0.67	0.75	0.77	0.68	0.70
HSV	0.78	0.83	0.78	0.93	0.68	0.77	0.95	0.63	0.75
$C_1C_2C_3$	0.72	0.89	0.78	**0.94**	0.62	0.73	0.77	0.71	0.73
$L_1L_2L_3$	0.74	0.92	0.81	0.70	0.78	0.74	0.76	0.69	0.70
$I_1I_2I_3$	0.87	**0.94**	0.89	0.91	0.75	0.81	0.91	0.67	0.77
HSI	0.87	0.92	0.89	0.91	0.74	0.80	**0.95**	0.65	0.76
YIQ	0.87	0.93	0.89	0.90	0.83	0.85	0.91	0.69	0.77
RGB	0.83	0.92	0.87	0.87	0.89	0.88	0.94	0.65	0.76
YUV	0.85	0.92	0.88	0.87	**0.89**	**0.88**	0.92	0.70	0.78
$SIFT$	**0.93**	0.90	**0.91**	0.89	0.81	0.85	0.94	**0.80**	**0.86**

Table 4.8: The criteria performances of CAH and SIFT feature using SVM classifier for Chars74K, TSIB, and ICDAR2015 datasets.

Color space	Chars74K			TSIB			ICDAR2015		
	p	r	f	p	r	f	p	r	f
$O_1O_2O_3$	0.63	**0.96**	0.73	0.84	0.75	0.77	0.95	0.59	0.72
HSV	0.79	0.78	0.77	0.87	0.74	0.78	0.82	0.63	0.71
$C_1C_2C_3$	0.63	0.57	0.58	0.90	0.67	0.75	0.97	0.60	0.74
$L_1L_2L_3$	0.77	0.76	0.77	0.84	0.67	0.71	0.80	0.70	0.71
$I_1I_2I_3$	0.83	0.82	0.81	0.93	0.65	0.74	0.88	0.66	0.75
HSI	0.80	0.79	0.79	0.93	0.63	0.75	**0.98**	0.61	0.74
YIQ	0.86	0.80	0.82	**0.95**	0.64	0.76	0.95	0.61	0.74
RGB	0.81	0.73	0.76	0.93	0.68	**0.78**	0.93	0.64	0.75
YUV	0.86	0.80	0.82	0.94	0.68	0.77	0.95	0.60	0.73
$SIFT$	**0.92**	0.84	**0.87**	0.57	**0.91**	0.70	0.77	**0.84**	**0.78**

Experiment 4: According to the results of SIFT and CAH using 1NN in experiment 3, we apply the benefit of the CAH and SIFT classifiers to be the adjoint SIFT+CAH feature to evaluate text/non-text classification by counting the number

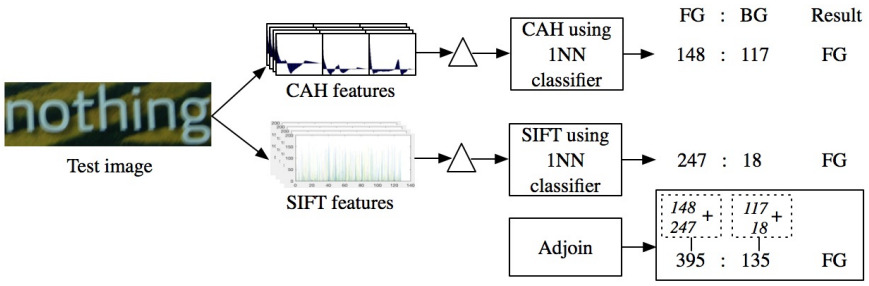


Figure 4.16: An example of FG and BG classification using 1NN classifier with both the CAH and SIFT features of the testing image are classified as text (FG) because CAH classifier presents the total number of correct matching for FG is 148 and 117 for BG, which the total number of FG is larger than BG. As well as SIFT classifier shows the total number of correct matching for FG is 247 and 18 for BG, which the total number of FG is larger than BG. The result of the adjoining feature for FG and BG is obtained from the total number of FG (148) and BG (117) by CAH classifier plus the total number of FG (247) and BG (18) by SIFT classifier (FG: 148 + 247 = 395, BG: 117 + 18 = 135), which the total number of FG is larger than BG. Therefore, the classification result for adjoin feature is text (FG).

of correct matching feature descriptors. Figure 4.16 describes the classification process. A testing image is extracted feature descriptors by the CAH features and SIFT features. The CAH features and SIFT features are classified as text (FG) or non-text (BG), respectively, by the CAH classifiers and Sift classifiers using 1NN. Then all classified features are counted as FG/BG. If the total number of FG is larger than BG, the image will be classified as text and vice versa. So, from Fig. 4.16, both the CAH and SIFT features of the testing image are classified as text (FG) because CAH classifier presents total number of correct matching for FG is 148 and 117 for BG, which the total number of FG is larger than BG. As well as SIFT classifier shows total number of correct matching for FG is 247 and 18 for BG, which the total number of FG is larger than BG. The result of adjoin feature for FG and BG is obtained from the total number of FG (148) and BG (117) by CAH classifier plus the total number of FG (247) and BG (18) by SIFT classifier (FG: 148 + 247 = 395, BG: 117 + 18 = 135), which the total number of FG is larger than BG. Therefore, the classification result for adjoin feature is text (FG).

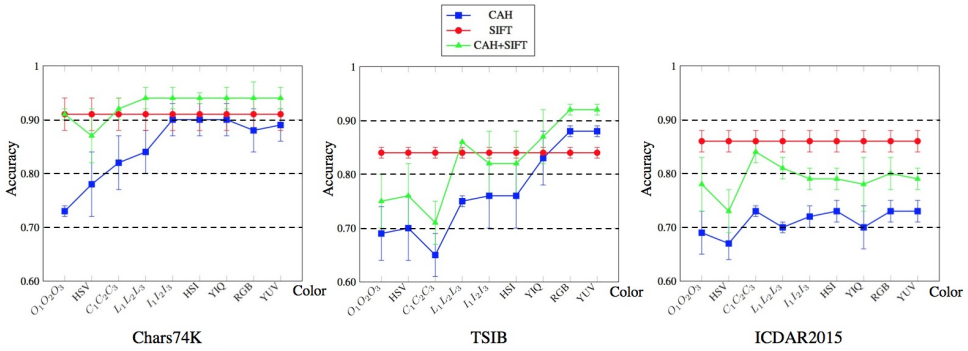


Figure 4.17: The accuracy of FG and BG classification using a 1NN classifier for the adjoint SIFT+CAH feature, compared for the three datasets. The results of the adjoint method confirm the usefulness of color (RGB or YUV) for Asian scripts Chars74K and TSIB.

The results in Figure 4.17 show that the proposed concept gives good classification accuracy (green graph) for the Chars74K and TSIB data sets. The accuracy with Chars74K presents a dramatic increase at 94% for other color spaces except $O_1O_2O_3$, HSV, and $C_1C_2C_3$. Furthermore, applying the adjoint feature to the TSIB dataset provides a better result for RGB and YUV at 92% of accuracy. Unfortunately, the SIFT feature outperforms the other features on ICDAR2015. However, we eventually find that when using a 1NN classifier for the adjoint SIFT+CAH feature, particularly color RGB or YUV, in order to classify text and non-text blocks for Asian scripts, Chars74K and TSIB is practical. Table 4.9 presents the comparison of adjoint features on the benchmark datasets by precision, recall, and f-measure. It shows that the proposed technique, i.e., the additional use of color, improves the performance, most clearly for TSIB.

Figure 4.18 shows the results of the experiments accurately classifying FG and BG of all three datasets. Although our proposed technique achieved a good accuracy for Asian script, it also has some observations with visual characteristics of images that can improve performance. In Figure 4.19, the image is classified incorrectly. For instance, there are some low contrast images on FG where the background and foreground color look similar. Some text blocks are struck by reflection, through which

Table 4.9: The criteria performances of adjoint feature for Chars74K, TSIB, and ICDAR2015 datasets.

Adjoint features	Chars74K			TSIB			ICDAR2015		
	p	r	f	p	r	f	p	r	f
$O_1O_2O_3 + SIFT$	0.86	0.95	0.90	0.94	0.72	0.80	0.90	0.76	0.81
$HSV + SIFT$	0.90	0.90	0.88	0.95	0.73	0.81	0.97	0.67	0.79
$C_1C_2C_3 + SIFT$	0.88	0.96	0.91	**0.96**	0.66	0.77	0.92	**0.79**	**0.85**
$L_1L_2L_3 + SIFT$	0.92	0.95	0.93	0.86	0.87	0.86	0.93	0.76	0.83
$I_1I_2I_3 + SIFT$	0.92	0.96	0.93	0.94	0.80	0.85	0.96	0.73	0.82
$HSI + SIFT$	0.93	0.96	0.94	0.94	0.79	0.85	0.98	0.69	0.81
$YIQ + SIFT$	0.94	0.95	0.94	0.93	0.86	0.89	0.95	0.75	0.83
$RGB + SIFT$	0.92	0.96	0.94	0.91	0.93	0.92	**0.98**	0.70	0.81
$YUV + SIFT$	**0.95**	**0.94**	**0.94**	0.92	**0.93**	**0.92**	0.96	0.74	0.83

both the CAH and SIFT classifiers cannot produce accurate results. In addition, the word image embedded in various color and cluttered backgrounds cannot provide a good result. Its feature descriptors are dominated by background features, and as a result, it has a chance to match BG keypoints in the codebook. Regarding a number of keypoints between FG and BG, if there are not many differences, it does not provide adequate information. Hence, the method could misclassify FG and BG. There are some BG blocks which have explicit color distinction after being extracted by SIFT. There are many keypoint descriptors. Therefore, the possibility of matching keypoints between the testing image and text keypoints in the codebook is increased. In the case of the background block with a logo, the background is filled in letters, and it is not a ground truth of text. When the block is an extracted feature descriptor, it could be matched to the descriptor of the FG codebook. So, this block may be classified as a foreground.

4.7.1 Comparison with State-of-the-art

To investigate the efficiency of the proposed method, we hence evaluated the performance of text and non-text classification compared to the other methods (different datasets and data sizes). The performance of Wu's method, tested with 84 test images, contained 367 text regions (minimum of three characters per region) of the ICDAR2003. Further, Alves and Hashimotos method also tested text and non-text classification with the same dataset as Wu's method. While, Sriman and et al. tested the experiments on the ICDAR2015 dataset, which 1,095 random FG zones and 1,035

Figure 4.18: Example of correct foreground and background classification on three datasets.

Figure 4.19: Example of missing foreground and background classification on three datasets.

random BG zones were used to evaluate the classification performance. For our proposed method used the ICDAR2015 dataset, which contains 1,943-word images and 1,943 random background images were used in the experiment. The experimental results in Table 4.10 show that the performance of the proposed method performs competitively against the results of existing methods in term of precision and recall.

Table 4.10: Text and non-text classification results for different methods.

Methods	Precision	Recall
Wus method (Wu et al. 2008)	78.87%	76.29%
Alvess and Hashimoto's method (Alves and Hashimoto 2010)	97.00%	88.00%
Sriman and et.al's method (Sriman and Schomaker 2015a)	94.89%	84.17%
Proposed method (adjoint feature)	**98.00%**	**70.00%**

4.7.2 Computational Complexity

This section presents the explanation about the computational complexity of the proposed algorithm through their execution time in order to determine the efficiency of the method. The advantage of this approach is the real estimation of the algorithm running time; however, it depends on the sample input instances, selection of functions for software development, as well as the employed hardware. There are four principal parts of the proposed algorithm: localization of point of interest, feature extraction, codebook histogram modeling and SVM classifier, and classification.

Localization of Point of Interest by Harris-Laplace detector (HL)

The interest points extracted by the HL detector are not only invariant to scale, rotation and translation, but also robust to illumination changes and limited viewpoint changes. To compute the complexity of HL detector is following this step.

- Create the scale space of an image f by a Gaussian function $P(\mathbf{x}, \sigma) = g(\mathbf{x}, \sigma) * f(\mathbf{x})$ where $\mathbf{x} = (x, y)$, which the complexity is O(nmp).

- For computing the subsequent detection of interest points at the scale, let $P_x(\mathbf{x}, \sigma_D)$, $P_y(\mathbf{x}, \sigma_D)$ is the derivatives at position $\mathbf{x}$ computed using the Gaussian filter of local scale σ_D. Then these intermediate results are composed to obtain three images $P_x^2(\mathbf{x}, \sigma_D)$, $P_y^2(\mathbf{x}, \sigma_D)$, $P_x P_y(\mathbf{x}, \sigma_D)$. These images are filtered by a Gaussian window size $\sigma_I(g(\sigma_I))$. Then, the single scale Harris corner indicator at position $\mathbf{x}$ in the image f is computed with $P(\mathbf{x}) = det(\mu(\mathbf{x}, \sigma_I, \sigma_D)) - \alpha trace^2(\mu(\mathbf{x}, \sigma_I, \sigma_D))$. At each scale, calculate local maxima for a corner, which gives runtime O(n^2). So, the complexity of this stage is O(nmp) + O(n^3) + O(n^2) $\approx$ O(n^3).

- Selecting the image blob characteristic scales, and the normalized LoG kernels with image scale space are defined by $|LoG(\mathbf{x}, \sigma_n)| = \sigma_n^2 \, |P_{xx}(\mathbf{x}, \sigma_n) + P_{yy}(\mathbf{x}, \sigma_n)|$. So, the runtime is O($n$) + O($nmp$) $\approx$ O(n).

- Salient positions in the scale space are detected by computing a multi-scale stack of the Harris corner indicator $\{H(\mathbf{x}, \sigma_k, \gamma\sigma_k)\}_{k=1}^{n}$, γ is a scalar (O(n)). For each scale σ_k, the local maxima of the Harris indicator $H(\mathbf{x}, \sigma_k, \gamma\sigma_k)$ are computed by $\{(\hat{\mathbf{x}}, \sigma)\} = argmaxlocal_{\mathbf{x}} H(\mathbf{x}, \sigma_k, \gamma\sigma)$. Then, select points at which the normalized LoG is maximal across scales and the maximum is above a threshold ($\hat{\sigma} = argmax_{\sigma} \, |LoG(\hat{\mathbf{x}}, \gamma\sigma)|$). The spatial location $\hat{\mathbf{x}}$ which do not hit a scale maximum, is given up. So, the runtime is O(n^2).

Feature Extraction

1. *SIFT feature*: There is the research to propose analyzing SIFT computational complexity (Drews et al. 2011), which presents into five stages: space scale extrema detection, keypoint localization, exact localization, keypoints descriptors and orientation, and matching. However, the matching stage does not employed in this research. The first stage is to create a scale space by the original image is generated in several octaves. Each octave's image size is half of the previous one, and the individual images are progressively blurred out using the Gaussian blur operator. In computational terms of the Gaussian blur by applying a series of single-dimensional Gaussian matrices in the horizontal direction, then repeating the process in the vertical direction is O($m_{kernel} m_{image} n_{image}$), where m is width and n is height. The computation complexity of this stage is $T(m, n)$ = O($mn - (mn/min(m, n))$), where m and n represent the width and height respectively in one image.

The second stage is keypoint localization by using the blurred images from the previous stage, to be computed the extrema (min and max) local values for each difference of Gaussians $D(x, y, \sigma)$ over scale and space at each octave. Therefore, each pixel in an image is compared with their eight neighbors into the current image as well as compared with nine pixels in the above and below scales. This process is done for all octaves iteratively in order to obtain potential keypoint localization. The complexity of this stage is $T(m, n)$ = O($mn - (mn/min(m, n))$).

The third stage is exact localization after the potential keypoints are found in the

previous step, which some of them were located on edge, or they have low contrast. So, these keypoints need to be defined by the Taylor expansion (Lowe 2004) to get a more accurate location of extrema, and if the intensity at this extrema is less than a threshold value, the keypoint is eliminated. The complexity of this procedure is O(k), where k is the number of extrema found in the previous stage.

The fourth stage is keypoint orientation and descriptors. Currently, the scale of detected keypoints is invariance. The rotation of the keypoints should be assigned to have invariance as well by computing the prominent orientations $\theta(x, y)$ and magnitudes $m(x, y)$. The $m(x, y)$ and $\theta(x, y)$ is calculated for all pixels around the keypoint of filtered image L with big O notation for $\theta(x, y)$ is O(1), and $m(x, y)$ is O($n^{\frac{1}{4}}$). Then a histogram is created to have 36 bins covering 360 degrees of orientation for the pixels in the region. It is weighted by gradient magnitude and gaussian-weighted circular window with σ equal to 1.5 times the scale of keypoint.The highest peak in the histograms correspond to dominant directions of the local gradient and any peaks above 80% of the highest peak are converted into a new keypoint, which has the same location and scale as the original, but different directions. After the keypoint orientation definition, the keypoint descriptor is created. The descriptors are computed using the gradient magnitude and orientation around the keypoint. To do this, there are 16 windows, and each window is broken into 4×4 subregion. For each subregion computes gradient magnitudes and orientations to have eight directions, which are put into 8 bin histogram. Then the 16×8 = 128 dimensional descriptors of this keypoint is completed. The time complexity in big O notation of this stage is O(k), where k is the number of keypoints.

2. *CAH feature*: This feature is the combination between an autocorrelation function and a color space. The input image (I) of creating the CAH feature is the cropped area image of a POI with the patched size received from Experiment 1. For each input image $I = 1toK$, loop through all POI of an image, is generated to three color level histogram of a color space (O(n) + O(3)). Next, the three color histograms are applied by autocorrelation function and they are adjoined to be the CAH feature (O($4n$)). So, the complexity time of the CAH feature is O(n).

Codebook Histogram Modeling and SVM Classifier

1. *Codebook Histogram Modeling*: In the clustering stage, we utilize the function `vl_kmeans` in the VlFeat implementation, is the k-means method of Lloyds algorithm (Lloyd 2006), to cluster CAH features and SIFT descriptors. Many parameters are used in the `vl_kmeans` function, i.e., initialization methods, number of time to restart k-means, and optimization algorithms. An approximated algorithm using approximate nearest neighbors (ANN) is one of a parameter in the optimization algorithms and suitable for huge problems, is set to the function. The ANN algorithm supports float or double data and uses Squared Euclidean distance (Eq. (4.19)) with time complexity in big O notation is O(n^2), to accelerate the sample-to-center comparisons. The following option for ANN algorithm is a k-d tree forest algorithm, which is used for containing the k-d tree indexing the matrix data and queries the index. The time complexity for the k-d tree in the worst case is O(n) (for all space, search, insert, and delete algorithm). The initialization method for determining the cluster centers or the means of the corresponding data points is k-means++ (Arthur and Vassilvitskii 2007), a popular method to avoid poor clustering found by the standard k-means algorithm. The k-means++ selects k centers arbitrary at random from the data points to initialize the cluster centers. The algorithm obtains O(log k) competitive that is guaranteed to find the optimal k-means solution.

$$\sum_{i=1}^{d}(x_i - y_i)^2 \tag{4.19}$$

2. *SVM Classifier*: The time complexity of creating SVM classifier is described as follow.

- For I = 1 to J, loop through all FG train images (O(n)) and BG train images (O(n)). Each image is extracted feature (O(n^2)). So, this get O(n) + O(n) + O(n^2), which means (O(n^2)).
- The codebook and train image keypoint descriptors are normalized to the range [0,1], which requires O(n).
- The image keypoint descriptors match to the codebook by computing distance (city block) between them to get a smallest pairwise distance and the

matrix I contains the indices of the observations in the codebook (ncb) corresponding to the distances in D, using pdist2 function in MATLAB ($[D, I] = pdist2(ncb, des, 'cityblock', 'Smallest', 1)$), where des is the descriptor matrix of train image (O(n)).

- Create FG train image histogram and BG train image histogram, which has a dimension equal to the codebook size, from the smallest distances derived from the previous step. For each dimension of the matrix I (O(n^2)), add 1 to the matched index in the FG histogram (O(1)) and repeat this step to the BG histogram. So, the complexity is O($2n^2$) + O(2) ≈ O(n^2).

- The FG histogram and BG histogram from the previous step are used to make SVMModel using fitcsvm function, and to make ScoreSVMModel using fitSVMPosterior function. The runtime is O(1).

Classification

1. *Nearest Neighbor (1NN)*: The time complexity of the 1NN algorithm with the distance function named pdist2 in MATLAB is described as follow.

- For I = 1 to J, loop through all FG and BG testing image observations (O(n)). Each image is extracted feature (O(n^2)). So, this get O(n) + O(n^2), which means (O(n^2)).

- The codebook and image keypoint descriptors are normalized to the range [0,1], which requires (O(n)). Determine the position of FG descriptors and BG descriptors in the codebook by dividing the codebook size into two parts using the expression, $sepPointCB = size(ncb, 1)/2$, where ncb refers to the codebook metric, $sepPointCB$ refers to the half position number of the codebook (O(n)).

- The image keypoint descriptors match to the codebook by computing distance (city block) between them to get a smallest pairwise distance and the matrix I contains the indices of the observations in the codebook corresponding to the distances in D, using pdist2 function in MATLAB
($[D, I] = pdist2(ncb, des, 'cityblock', 'Smallest', 1)$), where des is the descriptor

matrix of test image. Then, count the total number of the indices I to decide the descriptor is text or non-text, if $firstHalf = sum(I <= sepPointCB)$ the descriptors are located in the first half of the codebook that means these descriptors are text. If $secondHalf = sum(I > sepPointCB)$ the descriptors are located in the second half of the codebook that means these descriptors are non-text. So, this step gets runtime O(n) + O(1) ≈ O(n).

- To make a decision, if $firstHalf > secondHalf$ that means this image is text and vice versa, which the runtime O(1).

2. *Support Vector Machine (SVM)*: The time complexity of the SVM classification is described as follow.

- For I = 1 to J, loop through all test image observations (O(n)). Each image is extracted feature (O(n^2)). So, this get O(n) + O(n^2), which means (O(n^2)).

- The codebook and image keypoint descriptors are normalized to the range [0,1], which requires O(n).

- For each observation in the image descriptors, pdist2 function in MATLAB ($[D, I] = pdist2(ncb, des,' cityblock',' smallest', 2)$) finds the two smallest city block distances by computing and comparing the distance between observations in the codebook (ncb) and observation in the image descriptors (des). The matrix I contains the indices of the observations in the codebook corresponding to the distances in D (O(n)).

- Create test image histogram, which has a dimension equal to the codebook size, from a distance ratio comparison between two smallest distances derived from the previous step. For each dimension of the matrix I (O(n^2)), the first distance compares to the second distance times match ratio in order to analyze the keypoints that have been matched to point 1 and point 2 are different or not. If it is different distinctly, the keypoint is matched and add 1 to the matched index in the histogram (O(1)). But, if there are very similar, it is ambiguous that means the keypoint does not match. So, the complexity is O(n^2) + O(1).

- The histogram from the previous step is predicted by SVM classifier, which returns class labels and a matrix of scores (score) indicating the likelihood that a label comes from a particular class. The runtime is O(1).

4.7.3 Computation Times

The computation times of the proposed scheme for classification is elaborated in Table 4.11, 4.12 and 4.13, by depending on the optimal parameters received from Experimminent 1. The experiments are tested on the ICDAR2015 with total 3,498 train images (half each for text/non-text) for creating a model, and total 388 test images (half each for text/non-text). The minimum and maximum text sizes are 15×12 pixels and 1,735×833 pixels, respectively. For non-text, the minimum and maximum sizes are 6×4 pixels and 1,726×824 pixels, respectively.

Table 4.11: The execution time (in minutes) of the proposed scheme for SIFT classification using the 1NN classifier for the ICDAR2015 dataset.

Computation times	fold 1 (min)	fold 2 (min)	fold 3 (min)	fold 4 (min)	fold 5 (min)
Model creation	**310.06**	**369.10**	**341.03**	**371.49**	**295.31**
FG extraction	180.24	180.19	186.04	173.42	156.12
BG extraction	110.13	120.53	108.30	130.09	105.23
FG clustering	13.53	33.12	31.21	33.12	24.55
BG clustering	5.36	34.46	15.08	34.46	9.01
Classification	**12.00**	**17.48**	**9.02**	**15.23**	**45.12**
one image(avg)	0.031	0.045	0.023	0.039	0.116
Total process	**322.06**	**386.58**	**350.05**	**387.12**	**340.43**

Table 4.12: The execution time (in minutes) of the proposed scheme for SIFT classification using the SVM classifier for the ICDAR2015 dataset.

Computation times	fold 1 (min)	fold 2 (min)	fold 3 (min)	fold 4 (min)	fold 5 (min)
Model creation	**373.39**	**365.05**	**384.03**	**300.59**	**312.48**
FG extraction	210.10	202.19	197.37	155.34	164.36
BG extraction	129.15	128.58	138.43	118.15	118.18
FG clustering	13.12	14.07	25.03	14.20	15.20
BG clustering	21.02	19.41	22.40	12.50	14.34
Classification	**15.19**	**20.58**	**10.54**	**56.41**	**48.37**
one image(avg)	0.039	0.053	0.027	0.145	0.125
Total process	**388.58**	**386.03**	**394.57**	**357.40**	**361.25**

Table 4.13: The execution time (in minutes) of the proposed scheme for CAH+SIFT feature (the adjoint feature in different color space) classification using the 1NN classifier for the ICDAR2015 dataset in 5-fold.

Computation times (minutes)	$HSV + SIFT$	$YUV +$ SIFT	$HSI + SIFT$	$YIQ + SIFT$	$I_1I_2I_3 + SIFT$	$RGB + SIFT$	$L_1L_2L_3 + SIFT$	$O_1O_2O_3 + SIFT$	$C_1C_2C_3 + SIFT$
Model creation	**553.34**	**561.53**	**573.08**	**573.42**	**575.58**	**576.36**	**583.00**	**596.15**	**600.55**
FG extraction	309.25	313.10	323.13	318.00	323.24	323.01	328.34	335.20	338.42
BG extraction	179.40	183.06	187.15	191.11	186.27	186.24	188.48	195.22	192.06
FG clustering	45.43	47.15	44.37	46.18	47.03	47.45	46.39	46.23	50.03
BG clustering	18.46	18.22	18.03	18.13	19.04	19.26	18.59	19.10	20.04
Classification	**56.08**	**53.17**	**56.29**	**56.42**	**56.34**	**56.23**	**56.29**	**58.00**	**56.03**
One image(avg)	0.145	0.137	0.145	0.145	0.145	0.145	0.145	0.149	0.144
Total process	**609.42**	**615.10**	**629.37**	**630.24**	**632.32**	**632.59**	**639.29**	**654.15**	**656.58**

The results of the computation times (in minutes) of SIFT feature classification based on the same measurements, using the 1NN classifier (Table 4.11) compared to the SVM classifier (Table 4.12), which computed in 5-folds can be described that classification by the 1NN classifier took as less processing time as the SVM classifier. Table 4.13 depicts the execution times (in minutes) of the proposed scheme for CAH+SIFT feature (the adjoint feature in different color space) classification using the 1NN classifier for ICDAR2015 dataset in 5-fold. The adjoint feature $HSV + SIFT$ provided the shortest execution times in total process at 609.42 minutes. It can be noted that the computation times could be affected by the size and texture of an image, i.e., feature extraction of text gives much execution times rather than non-text. Consequently, computing the iterations of the sample-to-center comparisons in k-means clustering also have the ability to increase or decrease the processing speed.

4.8 Conclusion

In this chapter, an algorithm for a robust text and non-text block classification is proposed. It is based on the optimal hybrid combination of two features, CAH and SIFT. An elaborate selection experiment was performed in order to find the optimal classification models. The CAH and SIFT features were extracted around points of interests (POIs) detected by the Harris-Laplace (HL) detector. The results shows that the HL provides more useful POIs than the traditional difference-of-Gaussians method and has therefore been selected to be used here. All the POIs or keypoints

on the image are collected for feature extraction process. We performed experiments with multiple sets of criteria to find the optimal parameters for generating the CAH and SIFT features. Extracted features were used as initial parameters for generating the classification models. Among the possible sets of parameter settings, we selected the optimal set based on the obtained accuracy, using statistical testing. Finally, we also retained the features that represent color information in an intensity-invariant manner, by using the autocorrelation functions (ACF) on color histograms of color intensity and achieved a satisfactory FG/BG classification of around 90% accuracy. The accuracy of FG/BG classification with a 1NN classifier using the various types of color spaces tested on three datasets shows that color schemes RGB and YUV perform well on the Chars74K and TSIB datasets, while the performance on the ICDAR2015 benchmark data set is slightly lower.

In addition, in general, feature extraction using the SIFT algorithm provides a good result in scale and rotation invariant. It improves the accuracy of the models when it is combined with the CAH features. In the classification process, 1NN and SVM are used. The experiments show that 1NN gave a better result than SVM with both the CAH and SIFT features. In addition, the computation times of classification using 1NN compared to SVM, which tested on SIFT feature and based on the same measurements appear that classification by 1NN is faster than SVM. Regarding text in different languages, using different color spaces on Asian text gives more efficient information than in Western script, and this improves the performance of the classification models. Therefore, the parameters of the models can result in different classification performance; for example, a small C value of the SVM makes a large margin while a big C make a smaller margin or the appropriate number of support vector also improves the accuracy of the models based on the experimental results shown in this chapter. Finally, it should be noted that the approach is highly convenient for additional, later classification by other methods, such as convolutional neural networks: once interesting text regions are detected and preclassified by a computationally efficient method as proposed here, further refinement can be realized by more demanding procedures that now only need to process a small fraction of a potentially large image, e.g., '8k' type image of 7,680×4,320 pixels, a format that will be increasingly prevalent in applications.

Part III

Putting it all together

This chapter is an adaptation of paper:

Sriman, B., Schomaker, L. and Pruksasri, P. – *"General Text-Chunk Localization in Scene Images using a Codebook-based Classifier,"* Proceedings of the 4th IIAE International Conference on Intelligent Systems and Image Processing, pp. 134-141, 2016.

Chapter 5

Text localization in scene images using a codebook-based classifier

Abstract

Text localization is a main portal to character recognition in scene images. However, the detection of text regions in an image is a great challenge, and many locating methods use a bottom-up scheme that consumes a relatively high level of computation to identify the text regions. Therefore, this chapter presents a novel text localization in scene images using a top-down scheme, with a faster computation. While benchmark datasets are usually based on segmented words, there are also script types where character strings are continuous, such as Thai. Models of general text-chunk and non-text have been produced using the SIFT algorithm. Furthermore, we present a codebook-based classifier, which significantly enhances the detection performance. The proposed method has been evaluated for the text-detection and classifying processes with the TSIB (continuous-word language). The results show that our proposed technique improves the text localization particularly for a continuous-word script (Thai) which is rich in ornaments. The proposed technique is also beneficial for locating the text of the ICDAR dataset even though there are many text lines in an image.

5.1 Introduction

Today, mobile devices significantly influence daily human life. They are not only used for voice communication but also serve multimedia such as video streaming and sending images to people. New challenges in image processing such

as face detection, visual landmark recognition for navigation, object detection, and text recognition then arise for the scene images that are captured by mobile devices. Text localization is one of the biggest challenges to be accomplished. Many algorithms have been proposed to conquer this challenge, for example, Conditional Random Field (CRF) (Pan et al. 2009, Pan et al. 2011), connected components (Ezaki et al. 2004, Koo and Kim 2013) and stroke width transform (Epshtein et al. 2010). However, they have not fully succeeded because of different difficulties shown in several studies (Liang et al. 2005).

Proposed text localization methods are usually based on Western languages, which are punctuated with dots, and all the words in a sentence are separated by spaces. Also, most Western scripts have no tone accents in a word, but some lowercases in English present a dot on the top (i, j) and some have a line that is drawn up/down (f, h, k, l, t, p, q, y, g). Existing localization methods perform quite effectively in detection but using them for continuous-word script such as Thai consumes a relatively high level of computation.

Thai sentences do not have spaces between words. Space is only used to separate sentences, or words and numbers or symbols. Also, Thai alphabet structures are different from other languages scripts because of their complex geometric shapes, including lines (vertical/horizontal/tilted), circles and curves. Some letters have crossed lines and branched points. Moreover, some of them look very similar to each other. Finally, a Thai word is composed of consonants, vowels, and tones that can be four vertical zone levels in the written line, which can be easy overlapped between the upper line and lower line. Then, this raises the question of how to find the letters of words in a line of a sentence.

Continuous-word script detection in natural scene images is becoming more interesting and challenging, especially in Asian countries. A model-based text detection is a potential technique to locate character strings in scene images. Object histogram classifiers or codebooks could be used in the text localization process. However, to achieve a good result, producing models and relevant detection methods are crucial to solving the problem.

Furthermore, in performing text localization, the results could be text and/or non-text regions. In case text is not detected, the challenge of how to classify the candidate regions as text or non-text is also very important. In the literature, Support

Vector Machine (SVM), a machine learning algorithm is widely used for classifying text or non-text images (Kim et al. 2004). Koo and Kim (Koo and Kim 2013) classified text/non-text regions by creating a text/non-text classifier based on normalized gray scale images (without binarization), and the T-HOG descriptor (Minetto et al. 2013) is a text classifier that is used to characterize single-line texts in an image. However, the accuracy rates of these two classifiers are not very high depending on the process of the model production. Therefore in this chapter, we present a new continuous-word script localization and classification method using object histogram classifiers. The proposed technique has been evaluated on the text-detecting processes of a Thai scene image dataset (TSIB) (Sriman and Schomaker 2015b). The results show that our proposed technique performs effectively in text localization in scene images, particularly for continuous-word scripts such as Thai.

5.2 Proposed method

5.2.1 SIFT codebook histogram modeling

Locating objects in a picture is often performed in a natural scene OCR process. The object expectancy approach is a potential technique to solve this challenge. Finding only an object that is expected in the image is fast, uses low computation, and is similar to human behaviour. Internal models of the expected objects are the key to locating the area of the objects. Therefore, a model of the object is needed. In order to detect text regions, text models in the picture must be created. To ensure that the text detection is accurate and can distinguish text from the background, we proposed creating two types of object model: pure text codebook and mixed objects codebook. The detail of producing the codebooks is described as follows.

(a) Text and non-text selection

To obtain a codebook, we first need to prepare its constituents. The text codebook will be created using the text objects while the non-text codebook needs the background objects from training images to be built up. Based on the annotation of all the text regions that occur in the images, we extract all the text zones in each image

using its annotated data while additional steps can extract the non-text zones. First, the equivalent area of non-text to the text area in each image needs to be calculated to make it balance. We count the number of all the text zones and multiply them by the average dimension (width and height) of all the text zones of each image. The calculated area is called 'an expected area'.

We next find and extract the largest non-text area that is greater than the expected area from the picture. Finally, we obtain an equivalent area of non-text from the picture. However, some pictures cannot provide the equivalent non-text area because it is too small or less than the expected area. We consider skipping all the text and non-text zones of that image. Eq. (5.1) and Eq. (5.2) describe the selection.

$$avg_t = (\sum_{i=1}^{m}(w_i * h_i))/m \tag{5.1}$$

$$x_t = avg_t * m \tag{5.2}$$

Given p, where images by $p = 1..n$ in which each p has $1..m$ text zones represented by t. Let w, h be the width and height of each text zone respectively. So, the avg is the average area of the text zones in an image. Next, x is the expected area of the non-text that results from the multiplication of avg and m. Performing this selection should ensure that we get equivalent text and non-text areas for producing the codebooks. From the above formulas, we depict the process to obtain the text and non-text areas as shown in Figure 5.1.

(b) Feature extraction

All the text (FG) and non-text (BG) zones from Section 5.2.1 (a) are considered the constituents, and their attributes (features) will be used to generate parts of the codebook. We choose the SIFT (Lowe 2004) algorithm for extracting the features from each zone because SIFT is well-known and widely used to extract the feature of the objects in a scene image. The SIFT features are also known as keypoints, that consist of the coordinate (x, y), scale, orientation, and 128 descriptors. Several parameters can be adjusted to obtain a robust keypoint, for example, steps per scale

(a) (b) (c)

Figure 5.1: (a) An original image from the ground truth. (b) The black rectangles are the text zone from the ground truth. (c) The expected non-text area of the image.

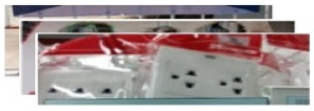
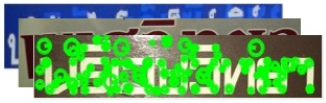
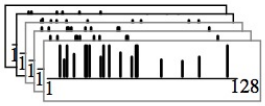
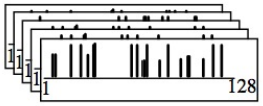
Training set	Training FG zones 1,850 images	Training BG zones 642 images
FG and BG blocks		
Compute POI by SIFT		
Extract feature by SIFT descriptor		

Figure 5.2: FG (1,850 images) and BG (642 images) blocks from the training set are used to compute POI (keypoint) by SIFT. There are many keypoints, and each image contains 128 dimensions of descriptors.

octave or initial Gaussian blur level (sigma). As a result, the SIFT algorithm generates text and non-text features from the zones (in Figure 5.2), and we combine all the text features to the first database as well as the non-text features to the second database.

(c) K-means clustering

The size of the keypoint database is typically large because of the unique key-

Algorithm 5.1 Pure text codebook (CB1)

Input: set of training text images, $Img = Img_1, Img_2, .., Img_m$;
Output: set of PKP of training text images,
$PKP = pkp_1, pkp_2, .., pkp_k$. Where m = number of total images
k = number of keypoints clusters (k = 4,000)
$Kps \leftarrow getDescriptor(Img_{1..m})$
$C = kmean(Kps, k)$
$PKP \leftarrow getCentroidC_{1..k}$

points location (x, y) but some descriptors are rather similar. The similarity of the descriptor may exist because the keypoints are generated from similar parts of objects. So, k-means clustering is applied to group similar keypoints to reduce the diversity. We found that grouping keypoints with a large number (k = 3,000) provided a high accuracy of the codebook (Sriman and Schomaker 2015b), but the computational time also increased when using too large a codebook. Based on this, we decided to group the keypoints k = 4,000, for which the computational time is acceptable. We employed k-means clustering (Forgy 1965) to partition the keypoints into 4,000 clusters for each database. Figure 5.3 illustrates receiving the clusters of the text and non-text, all the clusters centroids are called prototypical keypoints (PKPs) and these represent parts of the codebook. These PKPs produce two preliminary codebooks. The first codebook (CB1) in Figure 5.3 is the pure text, which consists of 4,000 PKPs of all the text zones as shown in Algorithm 5.1. However, during the experiment, using only the text codebook (CB1) could not give the necessary accuracy. So, we then try to combine PKP of text and background, where text PKP from 1-4,000 is the first half and BG PKP from 1-4,000 is the second half. These are called the second codebook which is shown in Figure 5.4.

(d) PKP improvement

Both the preliminary codebooks in Section 5.2.1 (c) consider whether they are text or non-text, based on the creation of text and non-text resources. In the second codebook (CB2), however, it is not guaranteed that the PKPs within the codebook are exact text or non-text PKPs. All the PKPs in the second codebook need to be improved to what they exactly are (text or non-text). For this reason, we apply an additional improvement to the second codebook by matching every PKP back to

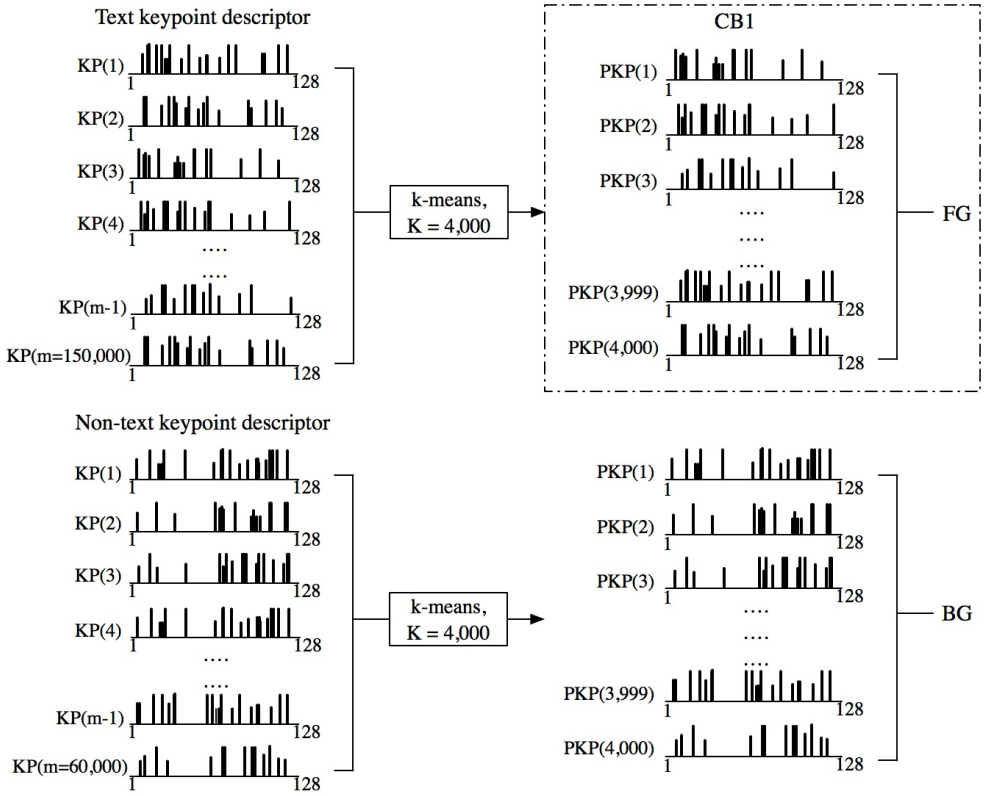


Figure 5.3: Because there are many numbers of keypoint descriptors of text and non-text, they will be grouped by k-means clustering, in which k = 4,000 of each text and non-text. The result of clustering is called the prototypical keypoint (PKP). By PKP_1 - $PKP_{4,000}$ (above in this figure) will be the first codebook or text model. The codebook is used for matching to the keypoint feature in an image, in which the matching keypoint could be a text object.

the text and non-text descriptors using the Euclidean distance based on SIFT. The matched number between the codebooks PKPs and the zones are represented as matching histograms, which is described in Algorithm 5.2. Figure 5.5 visualizes the matching process between text and non-text descriptors to CB2 in order to obtain the matching histogram of text and non-text.

Algorithm 5.2 explains the construction of text and non-text histograms. The

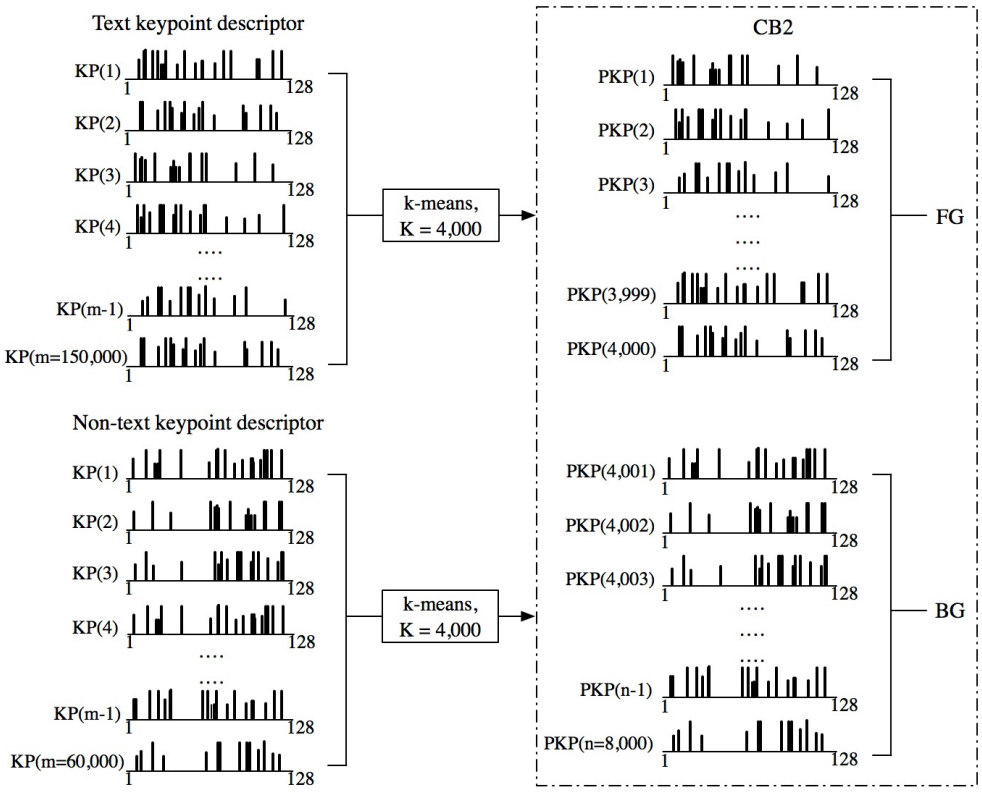


Figure 5.4: The second codebook (CB2) is the combination of text and non-text which consists of 8,000 PKPs (4,000 each) in total.

Algorithm 5.2 Improved codebook (CB3)

Input: set of text zones, $Txt = Txt_1, Txt_2, .., Txt_m$,
set of non-text zones, $NTxt = NTxt_1, NTxt_2, .., NTxt_n$,
the combined codebook (CB2), where m, n = number of text and non-text images, respectively.
Output: improved codebook (CB3).
$HisTxt = \sum_{i=1}^{m} match(CB2, Txt_i)$;
$HisNTxt = \sum_{i=1}^{n} match(CB2, NTxt_i)$;
$CB3 = sort(HistTxt - HisNTxt, desc)$;

histograms are generated from matching descriptors between CB2 to text (Txt) and non-text ($NTxt$) zones to get the two final histograms ($HisTxt, HisNTxt$). After

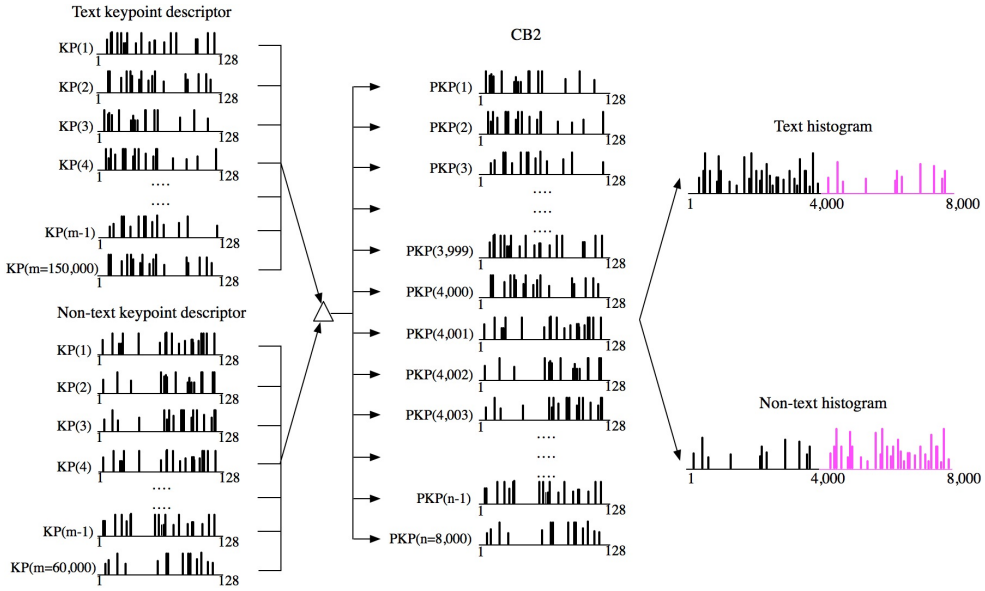


Figure 5.5: Construction of text and non-text histogram by matching text and non-text descriptors to CB2 using the Euclidean distance based on SIFT. The matched number between the codebooks PKP and the zones are represented as matching histograms. The histograms have 8,000 dimensions, of which the first half refers to text PKPs (black density), while non-text PKPs are in the second half (pink density).

that, we subtract the non-text histogram from the text histogram. The value of the subtraction could be positive or negative for each histogram bin. We consider that the positive index can confirm that the PKP is text because the number of matches for this PKP to the text zones is greater than to the non-text zones. Finally, we sort the subtracted histogram and assign the positive index as text and the negative index as non-text PKPs. The second codebook's PKPs are then rearranged into the new improved codebook (CB3).

The subtraction process is visualized to see the calculation distinctly as illustrated in Figure 5.6. Looking at the text and non-text histogram in the above rectangle, there are some PKPs that match to both the text and background. In order to get the appropriate PKP that represents the text or non-text exactly, the two histograms are subtracted. The subtraction result shows the positive values are green that could

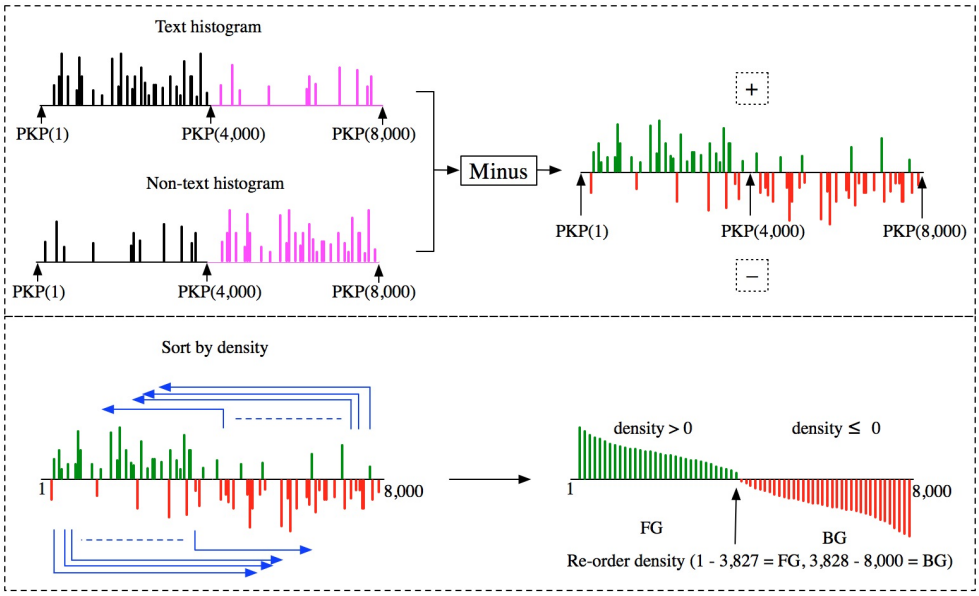


Figure 5.6: The subtraction of text and non-text histogram (in the upper rectangle). The lower rectangle describes the re-order of density, in which the left side is higher than 0 and the right is equal to or lower than 0. The position number 3,827 separates the left and right of which the left could be text and BG could be on the right.

be text, while the red refers to the minus values that could be non-text. Then they are sorted by density (in the bottom rectangle); the left side is higher than 0 and the right is equal to or lower than 0. The position number 3,827 is separated into left and right, of which the left could be text and BG could be on the right.

In the same way as the second codebook is re-arranged, it is sorted according to the consecutive position of the density. So, the improved codebook, called the third codebook (CB3), is then shown as the right rectangle. The PKP_1 - $PKP_{3,827}$ refers to text, the other PKPs refer to non-text as depicted in Figure 5.7.

(e) Generating the object classifier

Besides the codebooks, object verification models are also important to our proposed method. The verification model is a histogram model that can judge the detected candidate area as text or non-text. Both the text and non-text histograms

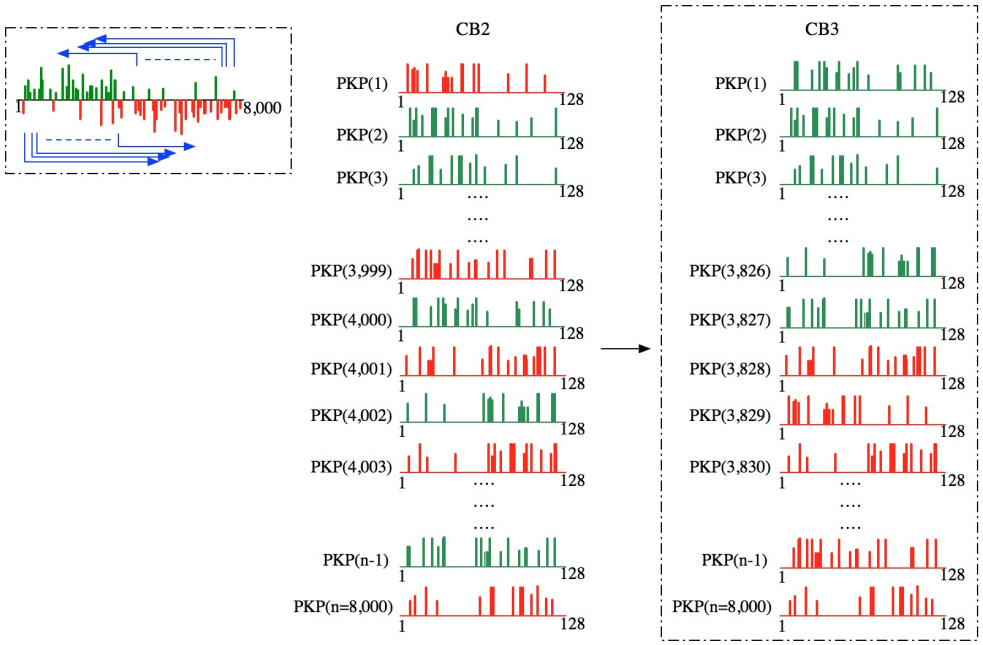


Figure 5.7: The re-arranged second codebook, which is sorted according to the consecutive position of the density. The PKP_1 - $PKP_{3,827}$ refers to text, and the other PKPs refer to non-text.

generated from the previous section (d) will be combined and normalized into 0-1. Using this process, we receive two verification models called classifiers as shown in Figure 5.8. Finally, we have 2 codebooks (text codebook (CB1) and the improved codebook (CB3)) and 2 classifiers (text and non-text classifiers).

To evaluate the accuracy rate of our classifiers, we test matching our codebooks back to the text and non-text zones of the training data sets and compute the distance between the matching result in histogram formation and the classifiers. Six distance algorithms are involved in the experiments: the Jaccard index, Dice coefficient, chi-square, Kullback-Leibler divergence, Bhattacharyya distance, and Manhattan distance. The shorter the distance, the more likely it is that the classifying zone is a member of that classifier. The accuracy rate of our classifiers is shown in Table 5.1.

- *Jaccard index*: The Jaccard index or Jaccard similarity coefficient was intro-

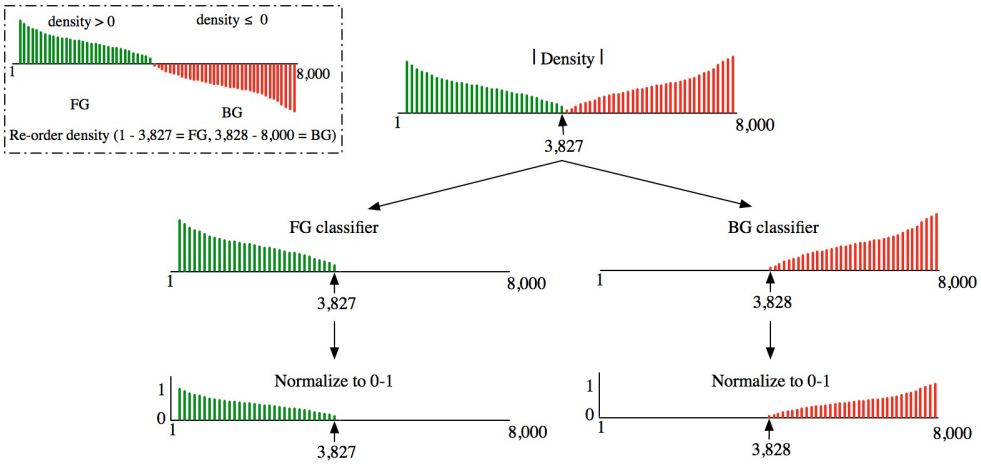


Figure 5.8: The process of creating text and non-text classifiers.

duced by Paul Jaccard in 1901 (Jaccard 1901) and is a statistic that is commonly used to measure the similarity and diversity between two binary vectors or small sample sets. The design of the algorithm is to see which members are shared and which are distinct as shown in Eq.(5.3).

$$\mathrm{D}_{x,y} = \frac{|X \cap Y|}{|X \cup Y|} \tag{5.3}$$

Given X, Y is a binary set. To compute the similarity between sets X and Y start with counting the number of elements which are shared between both sets. Then the total number of elements in both sets are counted. Finalize by dividing the number of shared members by the total number of members.

- *Dice Coefficient*: The Dice similarity index or Dice coefficient is used as a statistical validation metric to evaluate the performance of similarity coefficient between two binary samples as defined in Eq.(5.4). Where $|X|$ is the histogram of codebook, while the histogram of text/non-text images is referred to $|Y|$. The Dice similarity formula is started with the result of counting the number of elements which are shared between both sets multiplied twice and divided

by the sum of the number of elements in each set.

$$\mathrm{D}_{x,y} = \frac{2\,|X \cap Y|}{|X| + |Y|} \tag{5.4}$$

- *Chi-square*: The chi-square test, which was developed by the mathematician Karl Pearson (F.R.S. 1900), is used to describe the relationship between two categorical variables. In this case, the χ^2 is exploited to compute the independence between the training zones and the classifier. The chi-square test statistic is calculated by using the formula in Eq.(5.5).

$$\chi^2 = \sum \frac{(O-E)^2}{E} \tag{5.5}$$

Determine O is the observed frequency and the expected frequency under the null hypothesis is referred to E. The value of the test statistic is compared to the critical value of χ^2_α with the degree of freedom = $(r-1)(c-1)$, where r is the number of rows, and c is the number of columns. To reject the null hypothesis if $\chi^2 > \chi^2_\alpha$, ($\alpha = 0.05$).

- *Kullback-Leibler distance (KL)*: This is also known as the cross entropy, relative entropy in information theory. The algorithm, which was established by Solomon Kullback and Richard Leibler (Kullback and Leibler 1951), is an asymmetric measure distance that is used to discriminate differences between two discrete probability distributions. The smaller the relative entropy, the more similar the distribution of the two variables. The KL distance is defined by Eq.(5.6).

$$D(P \parallel Q) = \sum_{x \epsilon \chi} P(x) log \frac{P(x)}{Q(x)} \tag{5.6}$$

Given P and Q are probability distributions which have similar vector dimensions on a bounded set χ where $\chi \subset \mathbb{Z}$. Determine $P(x)$, $Q(x) \in \mathbb{R}$, where the

Table 5.1: Classification results and the timing information in millisecond per zone.

Distance method	Classification (%)	Ms/vector
Jaccard	94.29	70.88
Dice	94.23	69.88
Chi-square	94.10	74.50
Kullback	93.97	68.63
Bhattacharyya	93.59	69.25
Manhattan	93.39	67.63

elements of probability vectors $P(x)$ and $Q(x) > 0$, $\sum_{x \epsilon \chi} P(x) = 1$. Note that values P and Q are automatically normalized to the sequence as 1.

- *Bhattacharyya distance*: This is a divergence measure that is used to measure the similarity between distributions of features (Bhattacharyya 1943). The Bhattacharyya statistic is defined by Eq.(5.7).

$$D(P_1, P_2) = \int \sqrt{P_1(x) P_2(x) dx} \tag{5.7}$$

Let P_1 and P_2 represent probability distributions. Consider P_1 is similar to P_2 therefore D(P_1,P_2) = 1. Conversely, P_1 and P_2 are totally dissimilar, so, D(P_1,P_2) $\approx$ 0.

- *Manhattan distance*: The distance is measured by the x-axis and y-axis coordinates between two vectors (or points). For example, to compute the Manhattan distance (d_{ij}) between two points a = (x_i,y_i) and b = (x_j,y_j) which is defined as Eq.(5.8).

$$d_{ij} = |x_i - x_j| + |y_i - y_j| \tag{5.8}$$

The table shows that different distance measurements affect the accuracy of the classifier. The Jaccard, Dice and chi-square methods perform better than the others at 94.29%, 94.23% and 94.10%, respectively. Unfortunately, they are not so fast when compared to the Manhattan method at 93.39%, even though this produces a lower

classification result, and its time consumption in milliseconds per zone is faster than others by noise condition at 67.63 ms/vector. We therefore selected the Manhattan algorithm for our research.

5.3 Text localization

In this section, we present our proposed text localization method using the codebooks that are generated by the procedure in Section 5.2. Based on the expectancy approach, we expect that the text codebook should match all the texts in the testing image while the non-text model should match the complex background of the image. The visualization of the localization method is shown in Figure 5.9.

(a) Locating matched keypoints

The process starts by using the SIFT feature extraction method to extract the SIFT features (keypoint vectors) from a test image (Figure 5.9a). A set of extracted keypoint descriptors is matched to the prepared codebooks (CB1 and CB3). The matching process computes the distance between the keypoint descriptors of the image and the PKPs of the codebook (1-to-N). Using the Euclidean distance measurement algorithm for SIFT matching, all the computed distance results are then captured and sorted in order to find two minimum distances of the candidate PKPs. These candidates are next calculated to determine an explicit nearest candidate by multiplying the distance ratio (0.92) by the second candidate. If the distance of the first candidate is less than the multiplication, it is clearly distinct from the second one and it can be judged that the first PKP matched with the given testing key point. In contrast, if the first candidate's distance is higher than the calculation, these neighbors are considered too similar and ambiguous. We therefore ignore both the candidates PKPs. We repeat the matching for all the keypoints of the testing image. After the matching process, all the matched keypoints of the text are plotted by black circles with a radius equal to 15 pixels on a 2D spatial image (Figure 5.9b), which is exactly the same size as the testing image based on their coordinates (x, y).

(b) Dissolving patched object area

There are many matched keypoints scattered on the blank image. The plotted points represent the keypoints that are supposed to be the text in the image. To identify the detected area, merging adjacent keypoints together is the key. We pro-

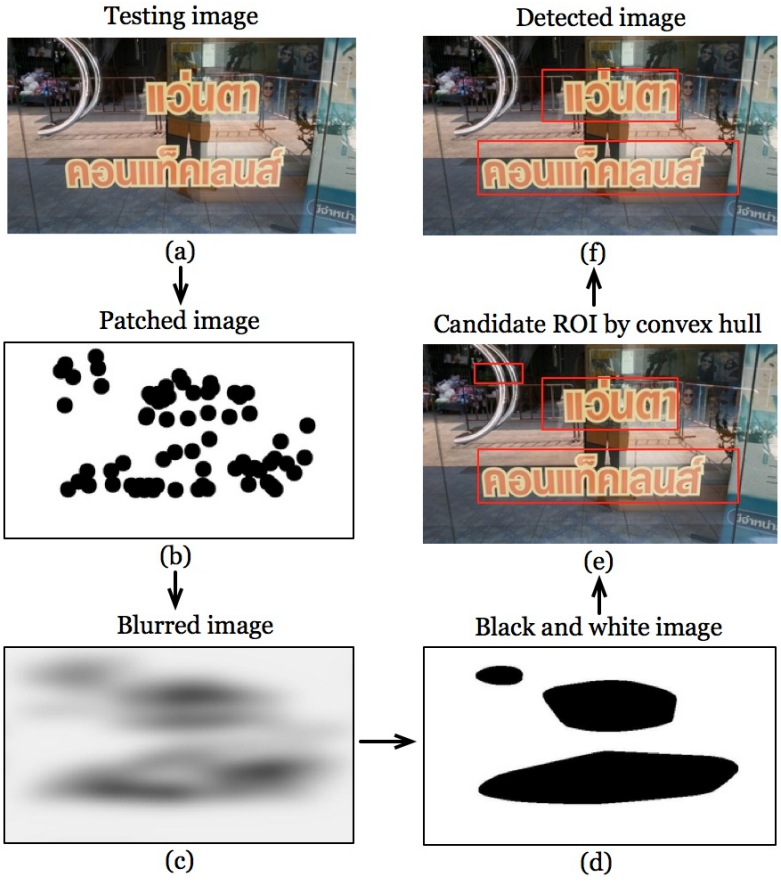


Figure 5.9: Detection process.

pose the dissolving method to do this. First, we apply a horizontal blur to the image and adjust the blurring width as a parameter. The different image dimensions give different blurring results when there is a specific width size ($width$) in the pixel unit. We then normalize the parameter to the percentage (10%) compared to the image dimension. The blurred points may be separated from others in the same text line. Therefore, the formula of the horizontal width normalization (hb) is $hb = width * 0.1$. Next, the fspecial function ($fspecial('motion', hb, theta)$) in Matlab is appropriate to the predefined 2-D filter image, the filter becomes a vector for horizontal and vertical motions. The angle of motion in degrees in a counter-clockwise direction is

defined by the theta, and the default theta is 0. This corresponds to a horizontal motion of nine pixels.

The stage of vertical merging should be performed. Next the Gaussian blur (Shapiro and Stockman 2001) in Eq.(5.9) is applied to the image. Here the distance from the beginning in the horizontal axis is referred by x, and y is the distance from the beginning in the vertical axis. All the separate lines will be dissolved to others by the standard deviation (σ) of the Gaussian distribution. The level of σ (e.g., 3.2, 4.8, 6.4, 8.0, and 9.6) can describe how blurred an image is. Figure 5.10 describes the different dissolving patched object area after applying the Gaussian distribution to a patch image by different σ thresholds. The higher value, the less blur there is. The lower value, the more blur there is. By this process, we receive the dissolved image (Figure 5.9c), in which the dark area is supposed to be the text objects.

$$g(x, y) = \frac{1}{2\pi\sigma^2} * e^{-} \frac{x^2 + y^2}{2\sigma^2} \tag{5.9}$$

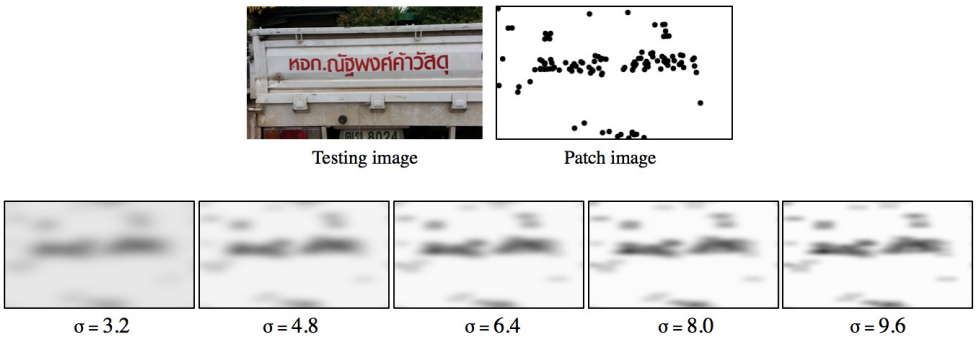


Figure 5.10: Examples of dissolving patched object area after applying the Gaussian distribution to a patch image with σ = 3.2, 4.8, 6.4, 8.0, and 9.6, respectively.

(c) ROI bounding box identification

At this point, an approximate location of the text is produced, but its area is still unclear. The dissolved image is then converted into black and white based on the normal threshold. The black and white image (Figure 5.9d) illustrates more accurate text areas in the picture. In many cases, the black areas are not linked to other text areas. To link them, a convex hull algorithm is applied with the function name's 'bw-

convhull' that was introduced in the Image Processing Toolbox in Matlab, in order to solve this problem. The convex hull formula is $CH = bwconvhull(BW,'objects')$. This estimates the convex hull of a 2-D logical matrix (BW) and returns the 2-D logical matrix (CH) or binary mask of all the foreground objects in the input binary image. The method $'objects'$ computes the convex hull of each connected component of BW individually, so CH will contain the convex hulls of each connected component. Then all the bounding boxes are extracted as the candidate region of interest (ROI) of the text. Figure 5.9e shows an example of candidate ROIs of the detecting method. Figure 5.11 depicts the examples of detection process and the candidate region of interest (ROI) of the text by the 'bwconvhull' function.

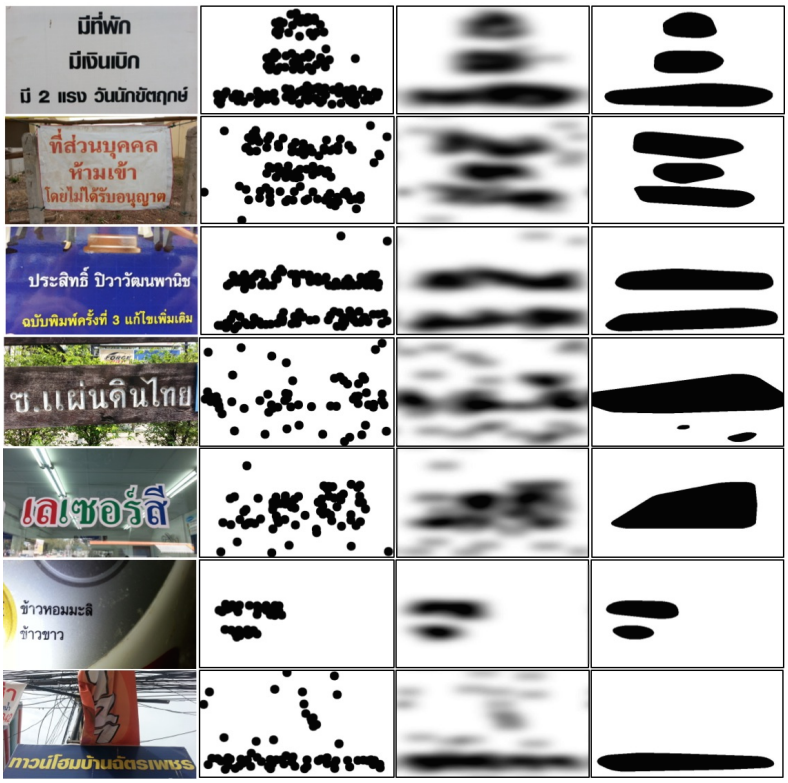


Figure 5.11: Examples of detection process and the candidate region of interest (ROI) of the text by the 'bwconvhull' function.

(d) Candidate ROI

As mentioned before, we already created the object classifiers for text and non-text. All candidate ROIs, which are extracted from the testing image, will be examined to make a final decision of its type with the classifiers. Figure 5.12 presents the examples of the candidate region of interest (ROI) before analyzing by the classifiers. The candidate ROI that is judged as text represents the final result of the localization while the ROI classified as non-text will be eliminated. Figure 5.9f shows an example result of our proposed method.

Figure 5.12: Examples of the candidate region of interest (ROI).

5.4 From text detection to character recognition

Given the detected text boxes in a scene image, the next step is to detect characters within such a box and sending them to a character classifier. Since characters in scene images have a much wider variation in appearance than those in regularly printed documents, the recognition module should be robust enough to deal with the variations. Also, the character extraction mechanism needs to be robust. There are currently many neural-network models for handling text recognition, such as the Long Short Temporal Modeling (LSTM) networks. This approach would definitely be useful for the current application. However, training may be costly and complicated. Given the fact that individual characters can be detected reliably, a

more basic segmentation approach can be interesting for the intended target hardware platform. Therefore, at the end of this dissertation illustrates the use of the text-detection method that was developed, a small application of the method will be shown. We will use an image-processing based approach for detecting the sequence of characters in a text box, sending each character candidate image to three common classifiers: (1) A regular multilayer perceptron; (2) a convolutional neural network and (3) the public-domain Tesseract recognizer (Thai-script version). The scale of this illustrative experiment is small. A more thorough implementation and testing of the subsequent OCR task is beyond the scope of the current project.

5.4.1 Learning model

The character models are built from 94 classes of Thai characters, Thai vowels, Arabic numbers, and symbols of the training set. We will test two classification approaches:

(a) The Multilayer perceptron (MLP) is developed from perceptrons, a probabilistic model for information storage and organization in the brain, invented by Frank Rosenblatt (Rosenblatt 1958). Rosenblatt was inspired to propose the use of layers of neurons as a computational tool by researching the model of the neural presented by the McCulloch and Pitts (McCulloch and Pitts 1943). The perceptron is a successful algorithm for pattern classification and pattern transform. The perceptron produces a single output based on multiple real-valued inputs by forming a linear combination using its input weights and then possibly passing the output through a nonlinear activation function. The original single-layer perceptron can be written mathematically as Eq.(5.10), where w is the vector of weights, x is the vector of inputs, b is a bias or activation offset parameter, and φ is the activation function.

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(w^T x + b) \tag{5.10}$$

A more general version, the multilayer perceptron, uses a supervised learning algorithm that is trained using error back-propagation. This means that there is a training set of input-output pairs, and the network must learn to model the depen-

dency between them. It learns a function $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where m is the number of dimensions for input, and o is the number of dimensions for output. A typical MLP network is composed of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. Figure 5.13 shows a single-hidden layer MLP with scalar output [1]. Given a set of training features $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where $x_i \in \mathbb{R}^n$ and $y_i \in 0, 1$. An MLP with one hidden learns the function $f(x) = W_{2g}(W_1^T x + b_1) + b_2$ where $W_1 \in \mathbb{R}^m$ and $W_2, b_1, b_2 \in R$ are model parameters. The weights of the input layer and hidden layer are denoted by W_1, W_2, respectively. The bias added to the hidden layer and the output layer are defined by b_1, b_2, respectively. For classification of output n classes, $f(x)$ passes through the activation function of a neuron, for instance the 'ReLU' where $f(x) = max(0, x)$. In classification problems, a one-hot vector encoding will be used: The target class element has a target output value of one, whereas the other elements of the output vector should return a value of zero. When a pattern is presented, the output vector will contain the likelihoods (pseudoprobabilities) that sample x belongs to each class (if $x \leq 0$, the probability of this class is considered to be zero). In the operational stage of processing, the class output is represented by the output unit that returns the highest 'probability'. More details on the standard MLP can be found in (Collobert and Bengio 2004).

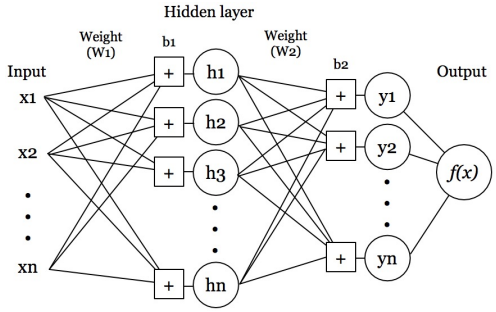


Figure 5.13: A multilayer perceptron with one hidden layer.

(b) The VGG16 model (16-layer model) is one of the state-of-the-art deep learning image classifiers in the Keras deep learning library presented by Simonyan and Zisserman (Simonyan and Zisserman 2014). Further, the model coefficients (weights &

[1] https://scikit-learn.org/stable/modules/neural networks supervised.html#complexity

biases) from a pre-trained VGG16 that is based on a large image data set is publicly available and can be used as a starting point for further training and fine-tuning.

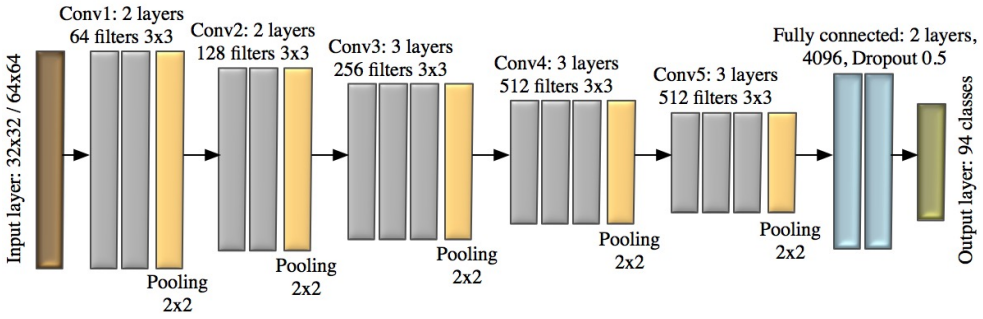


Figure 5.14: The VGG16 network architecture.

However, here, we will use its architecture but train the network, end-to-end, on the basis of our own data set of Thai characters. During training, the input to our ConvNets is the fixed-size of 64×64 and 32×32 RGB image. The model architecture (Figure 5.14) consists of five sets of convolution blocks with a max pool layer and a fully connected layer with 512 units that are activated by the ReLU activation function without data augmentation. The learning rate is set to 0.0005. The model outputs class probabilities based on the softmax activation function. During training, the Adam optimizer (Kingma and Ba 2015) and the categorical cross-entropy loss function are used.

5.4.2 Character extraction

In scene-text images, the text is often overlapped with other image elements. The characters showed in the text regions are colorful. Image blur and low resolution are typical. For these reasons, it is very difficult to extract the text regions from the background. Although current classifiers may be powerful when a correctly segmented object is presented in a cropped version that is comparable to what was seen during training, such a clean segmentation does not exist in case of a freely oriented (smartphone) camera. In order to facilitate character segmentation or selective attention to target objects in a large overall image, improving the image clarity is useful for the character image extraction stage. Standard image preprocessing methods do not

require extensive training and are able to remove unnecessary information. These methods include color reduction, converting a color image to gray scale, blurring of the image, image binarization, inverted image intensity, padding of the image, contouring of an image, application of the convex-hull detection and cropping image as described in Algorithm 5.3.

Algorithm 5.3 Character extraction

Input: cropped detected text region image
Output: characters of the input image
Let $D_{i=1..n}$ = set of n input images
for $Img = D_i$ **do**
$Img \leftarrow ReduceColor(Img, nums\_color = 4)$
$GImg \leftarrow RGBtoGray(Img)$
$GImg \leftarrow GaussianBlur(GImg, bsize = (5, 5), sigma = 0)$
$BImg \leftarrow Binarization(GImg)$
$numWhitePix = sum(BImg)$, where pixel = 255
$numBlackPix = sum(BImg)$, where pixel = 0
if $numWhitePix < numBlackPix$ **then**
$BImg[v == 0] = 255$
$BImg[v == 255] = 0$
else if $numWhitePix > numBlackPix$ **then**
$BImg[v == 0] = 0$
$BImg[v == 255] = 255$
$Img \leftarrow AddBorder(BImg, color = white, width = 10\%)$
$Co_{j=1..m} \leftarrow FindContour(BImg, RETR\_TREE, Approx\_method = Simple)$, where m is numbers of contour.
Let $Pc_{p=1..k}$ = set of possible character area
for $C = Co_j$ **do**
$Hull \leftarrow Convexhull(C)$
$ch \leftarrow DrawContour(Hull)$
$ar \leftarrow CalculateArea(Hull)$
$asp \leftarrow CalculateAspectRatio(Hull)$
if $0.5 \leq asp \leq 2.5$ **then**
$Pc_p = [ch, ar, asp]$
$Av \leftarrow AverageArea(Pc_{1..k})$
Let Ec = set of extracted characters
for $P = Pc_p$ **do**
if $(0.2 * Av) \leq P_{ar} \leq (4 * Av)$ **then**
$Ec = [Ec, Pc_p]$

(a) Color reduction: Color reduction is an essential technique to minimize the quantization error and produce a visually good result that can be applied as a pre-processing procedure for a complex color of a scene image. In this case, the number of colors of the test images is reduced to four, as presented in Figure 5.15.

(a) Original image color (b) Reducing image color

Figure 5.15: An example of color reduction.

(b) Converting a color image to gray scale: Converting a text image to gray scaleto reduce computation complexity, which is an easy and effective way in practice (Nikolaou and Papamarkos 2009). Gray scale can be good enough for recognizing certain objects. Since color images include more information for each pixel (i.e., 16/24 bits per pixel) than black and white images (0-255 per pixel), they can add unnecessary complexity and take up more space in memory. Figure 5.16 shows converting an image contrast to gray scale image.

(a) Reducing image color (b) Gray scale image

Figure 5.16: An example of converting color image.

(c) Blurring of the image: Making image smoothing is useful for removing noise from images by blurring with a Gaussian kernel (a low pass filter). This method is used with the getGaussianKernel() function provided by OpenCV, which using 5×5 box filter. An example result of blurring image is shown in Figure 5.17.

(a) Gray scale image (b) Blurring image

Figure 5.17: An example of image blurring.

(d) Image binarization: Then, the text region is produced the binarization to separate the character colors and the background colors by OTSUs algorithm (Otsu

1979). Through this process, a group of the candidate characters color and background color are represented. Figure 5.18 illustrates an example of image binarization.

(a) Blurring image (b) Binarizing image

Figure 5.18: An example of image binarization.

(e) Inverting image intensity: In our practices, making contours and crop character images by convex hull are sensitive to black font and white background. After obtaining image binarization, therefore, the foregrounds of all text regions are inverted to black color. The procedure that could check the text region is a black font or white font by computing the total number of black pixels and white pixels. In case the total number of white pixels is less than the total number of black pixels. The text region could deduce to have a white font; then, the white pixels are replaced to a black color and vice versa. Considering Eq.(5.11), determine v refers to a pixel value if v equal to 0, then the pixel value is replaced by 255. In contrast, the pixel is replaced by 0 when v equals 255. If the total number of white pixels is more than the total number of black pixels. So, the text region could have a black font, which the image does not have to be done anything.

$$BImg(v) = \begin{cases} 255, & v = 0 \\ 0, & v = 255 \end{cases} \tag{5.11}$$

(a) Binarizing image (b) Inverting image

Figure 5.19: An example of image inverting.

(f) Padding of the image: A part of a character in the image is sometimes located on the edge of the image, in which some parts are missing. When the black letters set on the edge are contoured, it will cover the areas that are all black. Then making

a convex hull on the contoured area, it gets the area that is too large and covers the entire image. To solve this problem, the padding image border, there will be white space that extends out of the black area. This makes it possible to contour the black character area. In the experiment, the original image is padded by a white border at 10% of the original image size. Figure 5.20 depicts the result of cropping character images on the original image size (a) compare to the padding image (b).

(a) Original image size (b) Expanding image size

Figure 5.20: An example of padding of the image padding.

(g) Contouring image: To this step, the shapes of the objects in the image need to be defined. Contours are a mostly preprocessing step to collect the region of interest in the image. It can be explained simply as a curve joining all the consecutive points (along the boundary), having the same color or intensity. After obtaining the image from the previous step, the image is calculated to find the contours and draw the contours using findContours() function and drawContours() function in OpenCV, respectively. There are three arguments in findContours() function that are source image, contour retrieval mode, and contour approximation method (CHAIN_AP_-PROX SIMPLE method shows only 4 points, which saves memory). The output of findContours() is the list of all the contours (each contour has (x,y) coordinates of boundary points of a shape with the same intensity in the image), which arranged by x-axis from left to right. Figure 5.21 shows an example of the contouring character image.

(a) Padding image (b) Contouring image

Figure 5.21: An example of contouring character image.

(h) Convex hull and cropping character images: After getting contouring images from the previous stage, the convexHull() function in OpenCV is applied to the contour position on the binary image to separate each character. To be assured that

(a) Contouring image (b) Convex hull image

Figure 5.22: An example convex hull computation to find character candidates in the image.

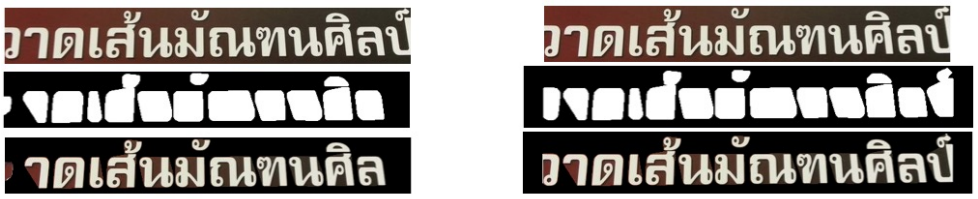
the contour is expected to be characters, the size of the convex hull is important. Then the convex hull area is calculated as well as the aspect ratio (asp = min(width, height)/max(width, height)) of every contour. If the convex hull has its aspect ratio between 0.5 and 2.5, then the contour, area, and aspect ratio of the convex hull are collected into the possible character array (P_c) and iterative this step for every convex hull. After getting all the attributes of possible characters obtained from the convex hull, they are computed the average area (A_v) to finalize the character image size of the text region. If the possible character area (P_{ar}) is between 0.2*A_v and 4*A_v, then it is realized to be a character image and collected into the expected character array (E_c). Figure 5.22 shows an example of the process of making the convex hull of expecting character image.

After obtaining the convex hull of all expecting character images, they are masked each individual filed contour with the original image to obtain the segments. The masked images are cropped into a rectangle image size, as presented in Figure 5.23.

(a) Masked image (b) Cropped character image

Figure 5.23: An example of cropping character image.

Figure 5.24 shows the comparison result of image extraction between (a) no padding image and (b) padding image. From Figure 5.24b, after the image is expanded its border, using the convex hull algorithm can calculate an expected character boundary more precisely. So, expanding the image border increases the number of character extraction in a text region.



(a) Original image size (b) Expanding image size

Figure 5.24: An example of padding image.

5.4.3 Character prediction with models

In this stage, the task is to predict the cropped character images obtained from the previous section. The labels of characters in the testing text regions are provided by human experts on the target and collected them in the ground truth. The ground truth is presented in JSON format, which consists of a list of image names. For each image, there are the following fields: text - the Thai alphabets presented in the image and unicode - the Unicode of each character as demonstrated in Figure 5.25.


```
{
  "name": "20140221_090650_1_zone.jpg",
  "text": "วาดเส้นมัณฑนศิลป์",
  "unicode": ["0xe27","0xe32","0xe14","0xe40","0xe49","0xe19","0xe21",
    "0xe31","0xe13","0xe11","0xe19","0xe28","0xe34","0xe25","0xe1b","0xe4c"
  ]
},
```


Figure 5.25: An example data of one image in the ground truth.

For label information, the original list is like [1,0,0,0,...]; because the final output should be a probability. The highest probability could imply the testing character image to be in that class. Figure 5.26 explains the probability result of a character image after the prediction process predicts it. The recognition result of a character image is presented as a Thai script. Figure 5.27 presents an example result of character recognition of a text region.

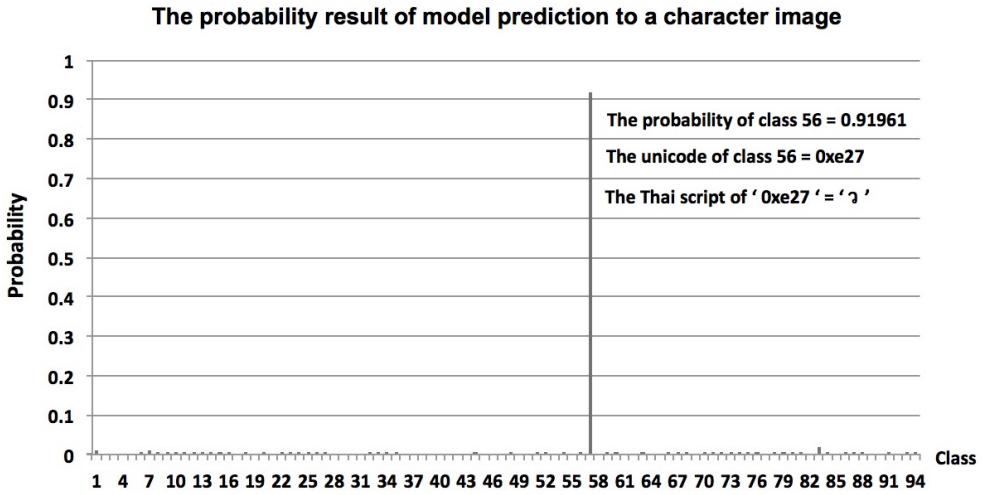


Figure 5.26: An example of classifier output (one-hot encoded vector) showing that the response to non-target character classes is well suppressed.

(a) Cropped character image (b) Character recognition

Figure 5.27: An example of a result of character recognition.

5.5 Evaluation and results

5.5.1 Dataset

In this chapter, we focus on improving general text-chunk localization in scene images using the proposed method. Image datasets are needed, and a Thai text dataset is selected to be used in this study. The dataset was already introduced by (Sriman and Schomaker 2015b), and is called the Thai scene image dataset, or in brief TSIB. TSIB consists of 1,566 images of scene images captured by a smartphone at 1,280x720 pixels. These images are divided into two groups for training and testing. All the text zones in each image are annotated as ground truth files. The training set is used for creating text and non-text codebooks, as well as the object histogram classifier. In order to reveal the performance of our improvement, the testing dataset is manually selected and separated into three sets. The single text line (TSIB-1) dataset

(a) Text zones

(b) Non-text zones

Figure 5.28: Examples of text (a) and non-text (b) zones.

consists of 403 images, and the multiple text line (TSIB-2) consists of 324 images. The combination of TSIB-1 and TSIB-2, named TSIB-3, consists of 727 images in total. According to the set of TSIB images, the zones of text-chunk are extracted. The number of the text and non-text resources shown in Table 5.2 and Figure 5.28 demonstrates examples of text (Figure 5.28a) and non-text (Figure 5.28b) zones.

Table 5.2: The text and non-text resources of TSIB dataset.

Type	Text chunk images			Zones	
	Single	Multiple	Mix	Training	Testing
Text	403	324	727	1,850	1,258
Non-text	-	-	-	642	590

5.5.2 Evaluation and results

(a) Text localization

We evaluated our proposed method using the definitions of precision and recall: $precision = |TP|/|E|$, and $recall = |TP|/|T|$, where TP is defined as the set of true positive detections while E and T are the total numbers of estimated rectangles and the area of ground truth, respectively. The F-measure is computed, based on the above precision and recall rate: $f = 2 * precision * recall/(precision + recall)$.

We prepare two codebooks (pure text and improved codebooks), as mentioned in Section 5.2.1. All the testing datasets are evaluated with codebooks with different parameters. We separate the testing into two steps. First, we perform the text detection with our proposed method. Second, we improve the detection using our

proposed classifier and then calculate the performances of precision, recall and the F-measure.

(b) Character Recognition

The dataset contained 1,573 candidate text regions from the text localization process, in which 400 text regions are selected for computing the character recognition experiments. The experiments are evaluated by calculating the precision, recall, and F-measure using the concepts of true-positive, false-positive, and false-negative. Here is the definition of these concepts and what they would be in this particular study:

The true-positive objects are correctly recognized characters by the MLP model and VGG16 model. That will be the characters that are presented in both the ground truth file and MLP's output file or the ground truth file and VGG16's output file.

The false-positives are incorrectly recognized characters as belonging to the positive class.

The false-negatives are all the characters that are in the image but did not get recognized, i.e., all the characters in the ground truth file that are not present in MLP's output file or VGG16's output file.

5.5.3 Results

(a) Text localization

Figure 5.29 shows the harmonic mean score of the detection from three experiments with different Gaussian thresholds (sigma = 3.2, 4.8, 6.4, 8.0 and 9.6) and fixed horizontal blur (10% of image width). The blue line is the detection result of the TSIB-2 dataset matched to the first codebook (CB1) with the non-classifier. The red line is the detection result of the TSIB-2 dataset matched to the improved codebook (CB3) with the non-classifier. The black line is the detection result of the TSIB-2 dataset matched to the improved codebook (CB3) with the classifier. The graph illustrates that the accuracy decreases when the Gaussian threshold is higher.

Table 5.3 presents the detection results of TSIB-1, TSIB-2 and TSIB-3 by matching them to the first codebook (CB1) with the non-classifier, the improved codebook (CB3) with the non-classifier, and the improved codebook (CB3) with the classifier, respectively. Figure 5.30 shows the detection result after using the classifier. The

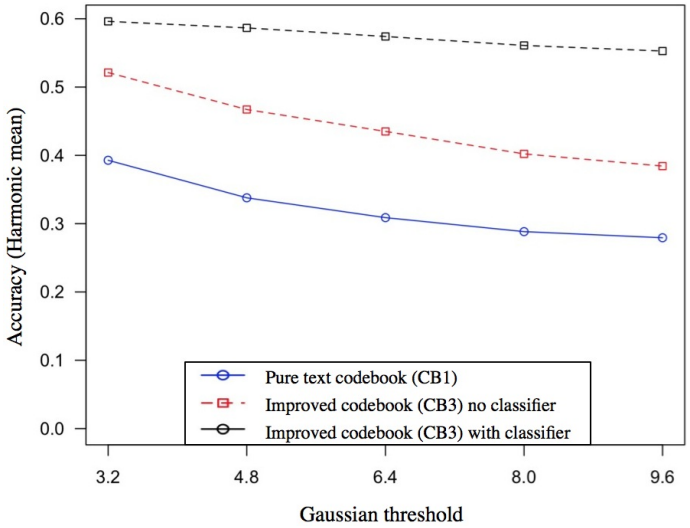


Figure 5.29: Harmonic mean curves for the three experiments with different Gaussian thresholds.

examples of the text detection results of the TSIB and ICDAR (Lucas 2005) datasets are depicted in Figure 5.31a and Figure 5.31b, respectively.

Table 5.3: The detection results of the three separated sets of TSIB with different methods shown as precision (p), recall (r) and f-measure (f).

Experiments	p	r	f
TSIB-1+CB1+no-classifier	0.30	0.75	0.43
TSIB-1+CB3+no-classifier	0.52	0.79	0.63
TSIB-1+CB3+classifier	0.71	0.79	0.75
TSIB-2+CB1+no-classifier	0.33	0.48	0.39
TSIB-2+CB3+no-classifier	0.56	0.55	0.55
TSIB-2+CB3+classifier	0.66	0.54	0.60
TSIB-3+CB1+no-classifier	0.32	0.57	0.41
TSIB-3+CB3+no-classifier	0.54	0.64	0.58
TSIB-3+CB3+classifier	0.69	0.63	0.66

Figure 5.30: Examples of text detection after applying the proposed classifier.

(a) TSIB

(b) ICDAR2003

Figure 5.31: Examples of text detection for (a) the TSIB dataset and (b) the ICDAR2003 dataset.

(b) Character Recognition

To obtain the model performances, it is evaluated by computing 5-folds cross validation for the MLP model and VGG16 model on the TSIB dataset. The training time, training accuracy, and testing accuracy are compared, which the results are obtained from MLP model structure using GridSearchCV function in scikit-learn (machine learning in Python), comprises of hidden layer sizes = (50,50,50), the activation function = 'relu', the optimizer or solver = 'adam', alpha (α) = 0.0001, and learning rate = 'constant'. In each fold of the training phase for all models, the input image size 32×32 and the epoch is set to 100 iterations. Table 5.4 displays the comparison results of training time of the two models.

Table 5.4: The training time result of the models.

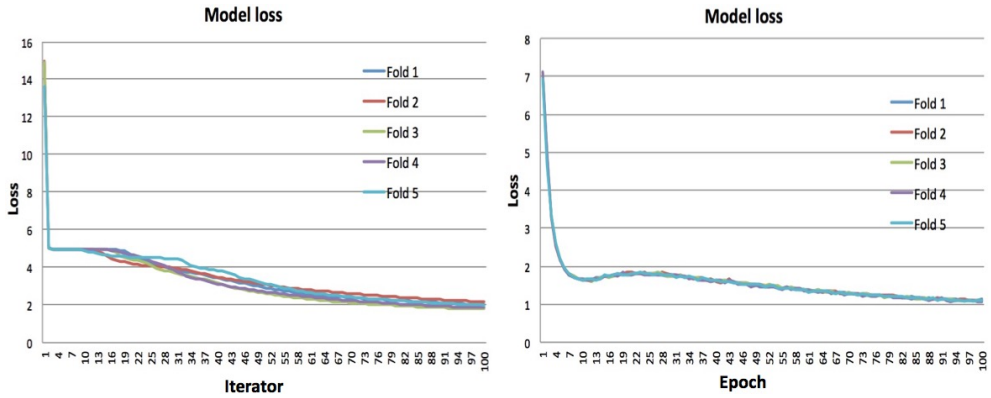
Training time					
Models	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
MLP	00:24:17	00:30:48	00:31:01	00:27:00	00:29:08
VGG16	07:51:41	07:49:48	07:53:38	07:59:45	07:41:08

From Table 5.4, we can see that the MLP with three hidden layers requires less minimal training time than the VGG16 for all cross validation folds. On average, the MLP with three hidden layers only needs less than 30 minutes to train. On the other hand, the VGG16 requires more time to train compared to the MLP due to the complex computations in convolution and subsampling layers. The model needs almost eight hours for the training phase, which are about 16 longer than the training time for the MLP model with three hidden layers.

Table 5.5: The accuracy result of the models.

Training accuracy					
Models	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
MLP	0.43	0.46	0.46	0.46	0.46
VGG16	0.96	0.97	0.97	0.97	0.97

Details of the training accuracy results of the two models are summarized in Table 5.5. Overall, the VGG16 outperforms the MLP in classification accuracy. This means that the convolutional and pooling layers in the VGG16 can successfully learn the relevant shape features of the text image. Further, the training loss curves of the two models in 5-folds are computed, which are shown in Figure 5.32. The Figures show that the plot of training loss decreases to the point of stability, is a good fit.



(a) The training loss curves of MLP model. (b) The training loss curves of VGG16 model

Figure 5.32: The visualization of training loss curves for the two models in five fold.

Figure 5.33 and Figure 5.34 illustrate the example results of character extraction and character recognition experiments. In the recognition process, the character images are recognized by the VGG16 model. Figure 5.33 presents character recognition that can be read as text; on the other hand, Figure 5.34 presents recognition results that are incorrectly interpreted as text. The incorrect recognition results are caused by limitations in the extraction processes, due to the complications in Thai script that are mentioned in Chapter 2. In addition, the characters are overlapping, which makes the character extraction even more difficult. Figure 5.35 shows examples of incorrect character extraction of the proposed method. There are two panels, left and right. Within each panel, the first column refers to the image ground truth, the second column refers to the label of the image ground truth, and the third column refers to the results of character image extraction.

Table 5.6 shows the results of character recognition (the character images received from the character extraction process) using MLP classifier and VGG16 classifier on the TSIB dataset that evaluated by precision(p), recall(r), and f-measure(f). Initially, the overall results of the VGG16 are better than MLP. The highest result provided by VGG16 (with parameters epochs = 200, input image size = 64×64), gives p, r, and f at 0.77, 0.76, and 0.75, respectively. After doing Pearsons Chi-squared test for MLP presents that p, r, and f are independent of epochs at X-squared = 0.0042, df = 8, p-value = 1. Similarly to p,r, and f of VGG16 that is independent of epochs at

Figure 5.33: Examples of character extraction and character recognition that can be read as text.

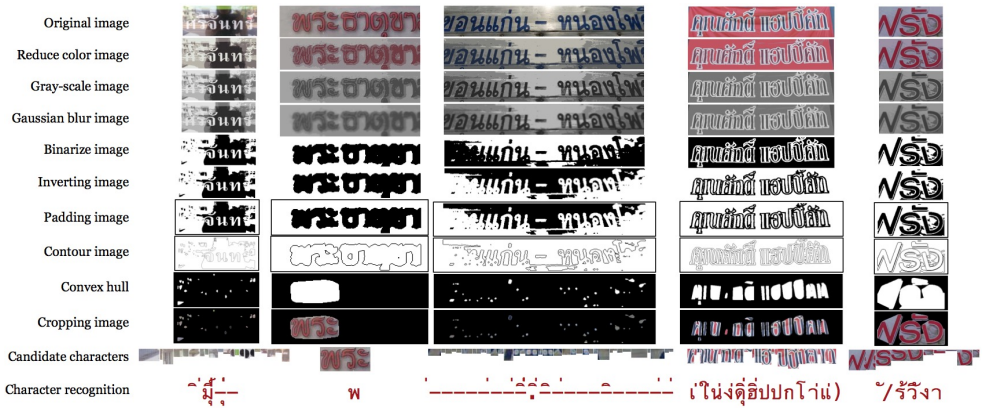


Figure 5.34: Examples of errors in character extraction and character recognition.

X-squared = 1e-04, df = 8, p-value = 1 with the same testing set.

The performance of the proposed systems is evaluated with the state-of-the-art (Tesseract) method. There are two important parameters of the Tesseract system used in the experiments: The Tesseract OCR engine mode is Neural nets (an LSTM engine) and a Page-segmentation mode extracting single lines of text from the image. The character recognition experiments performed with the Tesseract model are set to four kinds of the input text region images that we obtain from our own character extraction process: The original RGB images, the reduced-color images (four colors), the gray scale images and the padding images. The results reveal that the

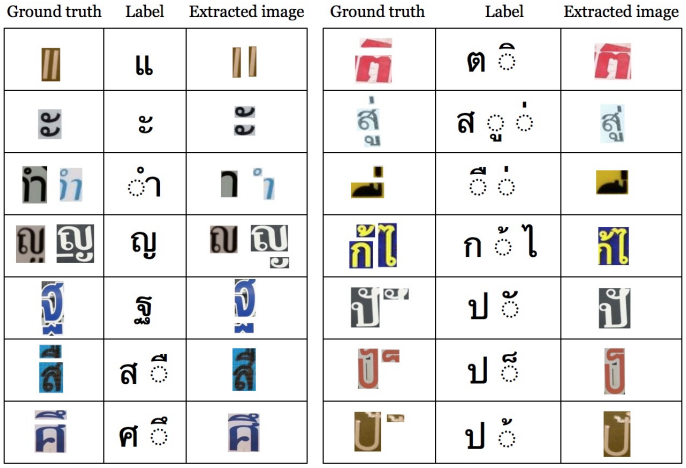


Figure 5.35: Examples of incorrect character extraction of the proposed method. There are two panels, left and right. Within each panel, the first column refers to the image ground truth, the second column refers to the label of the image ground truth, and the third column refers to the results of character image extraction.

Table 5.6: The results of character recognition using MLP classifier and VGG16 classifier compare to the Tesseract model on TSIB dataset that evaluated by precision(p), recall(r), and f-measure(f).

Classifier model for experiments	p	r	f
model_MLP (max_iter=100)	0.40	0.32	0.35
model_MLP (max_iter=200)	0.57	0.49	0.52
model_MLP (max_iter=300)	0.54	0.47	0.49
model_MLP (max_iter=400)	0.38	0.28	0.31
model_MLP (max_iter=500)	0.52	0.39	0.43
model_Tesseract (original RGB images)	0.63	0.61	0.61
model_Tesseract (reduced color images)	0.65	0.63	0.63
model_Tesseract (gray scale images)	0.66	0.64	0.64
model_Tesseract (padding inverted images)	0.75	0.76	0.75
model_VGG16 (epochs=100, image_width=32)	0.75	0.75	0.74
model_VGG16 (epochs=200, image_width=32)	0.76	0.75	0.75
model_VGG16 (epochs=300, image_width=32)	0.76	0.76	0.75
model_VGG16 (epochs=400, image_width=32)	0.76	0.76	0.75
model_VGG16 (epochs=500, image_width=32)	0.76	0.76	0.75
model_VGG16 (epochs=100, image_width=64)	0.76	0.76	0.75
model_VGG16 (epochs=200, image_width=64)	0.77	0.76	0.75

normal RGB images provide a lower recognition result than the other three kinds of

input images (reduced color images, gray scale images, and padding images), i.e., yielding just 0.63, 0,61, and 0.61 of p, r, and f, respectively. The inverted padding images achieve the highest recognition result at 0.75, 0.76, and 0.75 for precision, recall, and f-measure, respectively.

Although the Tesseract system is assumed to provide an acceptable recognition performance (Patel et al. 2012), there are still restrictions on the input images that are sent to the system. From the experiments it appears that the recognition results are degraded when the input images are (a) Distorted, contain reflections, or contrast problems; (b) Wrinkled and containing incomplete characters; (c) Contain shadows, and (d) contain a complex background, as shown in Figure 5.36.

(a) Distorted, contain reflections, or contrast problems

(b) Wrinkled and containing incomplete characters

(c) Contain shadows

(d) Contain a complex background

Figure 5.36: Examples of misrecognized text images by the Tesseract OCR engine, in LSTM mode.

5.6 Conclusions

In this chapter, we have presented our proposed method for text localization in scene images using a codebook-based classifier and character recognition. Thai text codebooks and classifiers have been generated and applied to the method. The text-locating performances show our proposed technique performs effectively. Although it works well with Thai text dataset, there are, however, some issues that can be further improved. First, even though the classifiers are simply generated from the histograms of the matching between the codebooks and training images without any intelligent algorithm, the performances are still good, but to improve the classifier, some machine learning algorithms such as Support Vector Machine (SVM) or Artificial Neural Network (ANN) can be applied. Second, our text-locating method using the calculation of bounding boxes from the convex hull generation causes inaccu-

rate ROIs in the text. The results of this codebook approach are promising, but color information is not used in SIFT. For future work, text and non-text classification on relative illumination independence using autocorrelation on color histograms, which exploits color, could be better than SIFT.

To quantify the performance of the classifier, k-fold cross validation was used to measure the classification accuracy and the training time. The same data are used for both the VGG16 and the MLP model. From the experiments, summarizing that the VGG16 model's accuracy is better than the MLP model's accuracy for the Thai scene-character recognition task, in all folds. However, the VGG16 model needs a longer time to be trained compared to the MLP model. From the experiments, it can be seen that the accuracy of the VGG16 model for the general (and clean) TSIB benchmark dataset can reach more than 95% in all folds. The performances of recognition of character extracted from text regions using the VGG16 classifier provide better results than using the MLP classifier in all character recognition experiments. The highest scene-text results are obtained by the VGG16 model with parameters: epochs = 200 and image width = 64, yielding performance of 0.77, 0.76, and 0.75 for precision, recall, and the f-measure, respectively. The Tesseract system only works well if the input images are well cleared. It is obvious that our proposed character extraction method combined with the VGG16 classifier outperforms the Tesseract model in terms of precision.

Chapter 6

Discussion

There are two main approaches for enabling a robot or intelligent perceptual application to emulate human vision in complex natural scene analysis: The top-down versus the bottom-up approach. The bottom-up approach entails looking for everything, everywhere, looking at every part of an image and then analyzing what the piece is. For example, a convolution window requires a recursive calculation of local features. This causes a high computational load. On the other hand, the top-down approach concerns a visual-attention mechanism that learns to focus on image regions that are expected to be relevant for the task at hand. The eye fixations in humans and animals are a good example of expectation-based visual sampling. In this approach, the system focuses on the big picture and from that derive the details to obtain evidence that supports the expectation. This approach, which is also similar to the model-based approach, defines what needs to be seen by comparing the object model to an object in an image. The idea of these model-based attentional patches can be used much more widely in real-world scene perception (Walther et al. 2004, Jian et al. 2015, Xiao et al. 2015, Foulsham and Kingstone 2017).

Following the attentional-patch principle, we used SIFT for the patch classification. A patch-based attention model using SIFT provides a descriptor for points of interest (POI) in the image. The presence of several SIFT key point descriptors is a good indicator of the 'textuality' of a region in the image. However, the results of object classification using only the SIFT patch-based model are not very good. For instance, in Chapter 2 at the stage of character recognition, a character feature extracted by SIFT in the grayscale image does not provide a sufficient number of keypoint for character recognition. So, to enlarge the number of keypoints, a character image is also extracted from a binarized (B/W) copy of the character image. There is the efficient approach that can increase the number of POI, is feature extraction by

the Harris-Laplace (HL) using the multiscale Harris cornerness measure, provides many number of POIs than the Difference of Gaussian operator in SIFT. Therefore, object classification using additional patch-based features could be improved.

The human color perception can be approximated by using a combination of three primary colors, which used to replicate color scenes in photography and other media. These primary colors are also used in many image processing applications for robot learning to be able to mimic the color vision as human beings. Color information is attractive to object classification and has a larger effect on visual identification (Vidakovic and Zdravković 2009). An achromatic image color, the colors may disappear after a single layer, and most information could be lost by definition. In contrast with color space, i.e, RGB image has 3 dimensions, it is impossible to derive 24 full bits of information from 8 bits. The studies of neuropsychological about brain-injured patients with impaired object recognition abilities (Humphrey et al. 1994, Mapelli and Behrmann 1997) have shown that identifying colored objects improved performance rather than black and white and grayscale images. Tanaka and Presnell (Tanaka and Presnell 1999) have introduced color diagnosticity by using colors to reference the specific characteristics of the object based on the color diagnosticity hypothesis. It explains that color information would affect the recognition of objects with high color diagnosis values but would not affect the recognition of objects with low color diagnosis values. Ito and Kubota (Ito and Kubota 2010) have mentioned that properly colored features are classification faster than monochrome features.

However, while retaining the structure with a vectorial organization (i.e., using mean and standard deviations of the expected positions for critical structural elements in Thai characters), the convolutional neural networks (CNN) could easily be applied to classify the individual patches. In this way, the method we proposed (Sriman and Schomaker 2015b) can be used in general computer vision, e.g., for robots. The following example (Figure 6.1) of the visual analysis of an excavator machine while a robot or AI agent is approaching it explains this idea schematically. The excavator machine has a relatively stable structure, for which recognition by parts is applicable. Even if a CNN is very good at classifying a complete machine by the typical low-resolution full image of 256×256 pixels, from a distance, a robot approaching the machine will need to be able to identify the individual parts: text

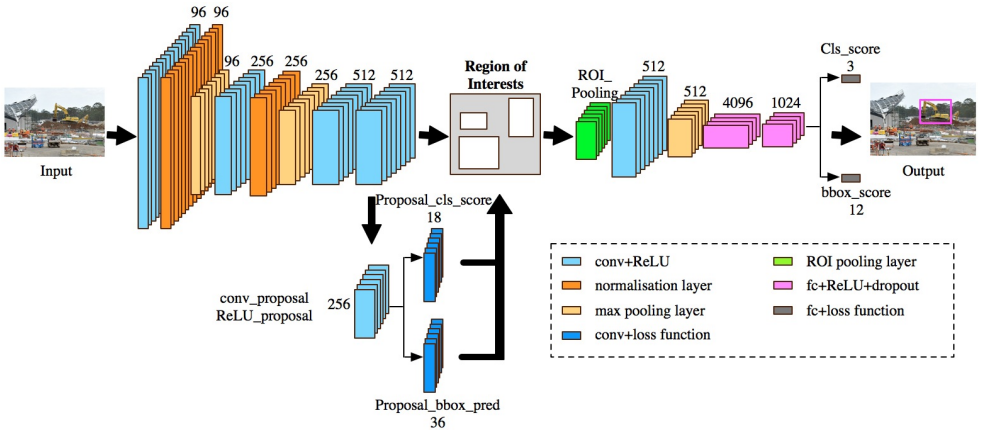


Figure 6.1: The network architecture of the modified detection model based on the framework of Faster R-CNN [redrawn after] (Xiang et al. 2018).

blocks, wheels, bucket, alarm light, etc. To understand the visual scene, the attention of a complete pretrained CNN will need to be directed at specific regions of the total high-resolution image.

At each point identified as CNN-attention x in Figure 6.2, the system would need to provide a classification. In this task, a statistical structure for where the essential elements are usually found will be helpful for such a robot. The location of some parts is more stable than that of others. For example, the distance between the wheels is fixed, and variations will be due to perspective. The bucket, on the other hand, maybe in several possible positions but even for this structural element, there will be a statistical regularity in its relative locations. The example illustrates the general applicability of the attentional patch concept. Another interesting real-time object detection in streaming video system called You Only Look Once (YOLO) (Redmon and Farhadi 2018) has been presented by Joseph Redmon and et al. It uses Deep Learning at 100 layers and processes images at 30 FPS, which is achievable in operation, but training stage is very slow. In the testing process, it makes predictions with a single network evaluation by predicting the position and size of the box region and classifying the class probability of the box at the same time. However, its region classification cannot handle huge numbers of classes, and It did not work

well in handwriting recognition.

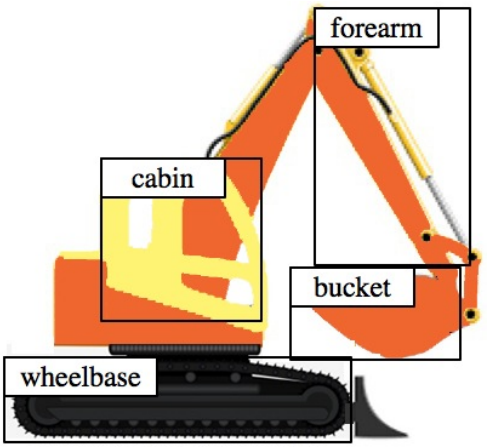


Figure 6.2: Example of initial positions of tracking boxes [redrawn after] (van Boven et al. 2018).

Recently, CNN or deep learning has evolved into a powerful technique used in image analysis, e.g., scene text detection, classifying objects, and object recognition. However, there is a sophisticated modeling process; for instance, multi-digit number recognition from Street View requires an astonishing number of 11 network layers (Goodfellow et al. 2013). Deep learning, therefore, requires not only a massive amount of labeled sample data but also substantial computing resources. In order to be less time-consuming, the size of the raw input image from the camera is usually reduced. Besides, deep learning architecture is very complicated that we do not know the working process within the algorithm. In contrary, the object attention patch manipulated by the principle of BOW is straightforward, which can be managed or defined the input data, assign various parameters by ourselves. The results acquired by this process are acceptable.

Furthermore, there is evidence from a handwriting recognition experiment (LeCun et al. 1995) that compares several learning algorithms, including CNN and SVM (support vector machine), which indicates that the optimal margin classifier using the polynomial kernel has excellent accuracy. In fact, this classifier would do just as well if the image pixels were permuted with a fixed mapping, while CNN would give a worse result. This indicates that SVM does not need prior knowledge about

the problem, and classifying by SVM can achieve a good result and less memory consumption than CNN. Another important point of SVM is the training process, which was set up as a cautious solution with a quadratically constrained optimization. The function can be easily applied to optimize various processes and be verified that we have the best value. Therefore, training by SVM is easier than CNN.

Smartphones, in recent, are packing more power, and some are overtaking desktop computers in speed and reliability. However, the training model on a smartphone often uses a computational heavy. Most deep learning models today train on specialized hardware like GPUs, which can compute large models or complex data for many days. This causes the training process usually happens in the cloud on powerful servers [1]. There is an open-source framework from Google called 'Tensorflow' for scientific and numerical computation. Tensorflow is not used to run on smartphones except eight pre-trained models taken from state-of-the-art optimized research models [2]. Using the optimized model to run on mobile would be faster than running on the server. However, it is impossible to use a custom model to run on mobile devices at this time because there is a problem with different smartphone resources, i.e., CPU and ram. It can be concluded that the character recognition using deep learning is not suitable for mobile computing, but still have to work with the processing on the server.

6.1 Answers to the research questions

Q1: How is it possible to indicate where text lines are in a natural scene image?

The difficulty of text detection in a complex background is usually solved by bottom-up schemes such as salience (edges) or the stroke-width of the objects. In a series of pilot experiments, the results present a lot of false positives or non-specific detection of text. Looking at human vision, the expectancy approach plays a central role in detecting objects in a visual scene. The top-down scheme, which can only find objects that are expected to be in the image, is fast, uses low computation, and is similar to human behavior. Regarding the text format that appears in a natural image, Western and Asian languages have different structures. Using a continuous-

[1] https://algorithmia.com/blog/machine-learning-and-mobile-deploying-models-on-the-edge
[2] https://www.tensorflow.org/lite/models/object_detection/overview

word script model localization in a natural scene is beneficial for Asian countries such as Thailand because the content is frequently in character strings, which do not have spaces between words. Therefore, internal models of expected objects using SIFT descriptors (a powerful feature to solve the problem of detecting images that are different in scale, rotation, viewpoint, and illumination) are the key to locating the area of the objects. In addition, the integration between motion blur and Gaussian blur is useful for identifying an approximate text location and finalizing the specific candidate region using a convex hull algorithm. The proposed technique performs effectively for Thai text dataset.

Q2: How can text and non-text blocks be identified?

Given an obscure image element, people can accurately distinguish it in the foreground (text) or background (non-text) with informative learning, for instance, text patterns and the surrounding objects. Creating a reliable feature model of text and non-text by human intuition and then training a robot with this knowledge would confidently discriminate a component as text. One simple and flexible model procedure is a bag of visual words (BOVW), which represents the occurrence of object attributes within an image: object description (a SIFT feature), color distribution (a CDis feature), and gradient strength (a GLAC feature). These three properties are utilized to describe the dissimilarity between text and the background. Then the object models are constructed from these categories collaborating with the SVM to generate the classifiers. The observation feature is judged by probabilistic voting weight to be text or non-text. The proposed method achieved promising classification results on the ICDAR2015 dataset.

Another technique for solving the distinction between text and non-text is the color autocorrelation histogram (CAH) adjoined with the SIFT algorithm. Due to a colorful natural scene, some color features seem to hinder the correct classification of text/non-text. Additionally, the illumination variation, shadows, and reflections within a natural scene, will influence the performance of color-based image classification. To relieve these problems, it was hypothesized that it would be possible to use autocorrelation to represent color information in an intensity-invariant manner, by using the auto-correlation functions (ACFs) for the histograms of color intensity in red, green and blue, or even in other color coding schemes. The model classi-

fier that adjoins the CAH and SIFT using 1NN performs robustly and is suitable for Asian scripts such as Kannada and Thai.

Q3: How can a class of scene image characters be recognized?

A character is a composite of several elements such as lines (vertical/horizontal /tilted/curves/circles), a series of uniform color regions, or a group of textures. Some letters have crossed lines and branched points. Text from various languages shows different characteristics. Considering English and Thai, English has 62 alphanumeric characters of single components while Thai has 44 consonants, 21 vowels, 4 tones, and other symbols. There is some similarity between the Thais consonants and Thais vowels. English has no tone accent symbols, but some lowercases in English present a dot on the top (i, j) and some have a line that is drawn up/down (f, h, k, l, t, p, q, y, g). Referring to the background, some text bears a resemblance to features such as fences, windows, or leaves that have a similar stroke, pattern, or edge properties, and this make it difficult to design adequate feature representation to identify text.

Characters are well-defined patterns and based on salient heuristics that often focus on the intensity, color, and contrast of objects appearing in an image. Saliency detection is a coarse textuality estimator at a micro scale, yielding the probability for each pixel that it belongs to the salient object (Borji et al. 2015), while the information such as luminance and color space (e.g., RGB) is of limited dimensionality. Hence, using a larger region at mesoscale, i.e., the size of the characters, increases the information used for the textuality. In this way, the expectancy of a character is modeled by attentional patches and exploiting the detection of small structural features by the SIFT method, in combination with modeling the expected 2D layout of these key points in characters. The process can handle variable sizes and fonts, and similar-shaped characters are substantially better recognized.

6.2 Further research directions

1) The opportunity to have an experience with Thai scene image dataset (TSIB)

Most published natural image datasets are based on European script (Lucas et al. 2005, Wang and Belongie 2010, Smith et al. 2016), and some datasets are in

Asian script such as Kannada (de Campos et al. 2009), Korean (Jung et al. 2011), Chinese (Yao et al. 2012), but not Thai script. So, we would like to introduce a new scene image dataset (Thai scene image dataset (TSIB)) that was taken by a smartphone to capture Thai script images that are ubiquitous on the street in four provinces of Thailand. The images were obtained in various general scenes and under different conditions including perspective, light, distance, and complicated background. This dataset contains 1,566 images in total together with the annotated characters and words in xml file. The characters contained in the TSIB dataset are diversity in shapes and label types: not only character codes but also Thai consonants/tones/vowels/symbols/numbers, and Arabic numerals. The majority of the pictures have a resolution of 1,280×720, and a minority of the pictures have resolutions of either 1,600×1,200 or 1,920×1,080. Thai script is different from European script and other Asian scripts (as has been discussed in Chapter 2, p. 30, section 2.3.2). This image dataset is challenging for text localization, word segmentation, character recognition as well as text translation.

2) The improvement of text localization

Text localization by the bottom-up method often presents false positives or non-specific detection of text because small patches may not have enough information for classification. The text localization algorithm proposed in this thesis relies on the principle of object attention patches that have made use of the models of general text-chunk and non-text to be the expected object. The model achieved the goal of detecting Thai script and is also beneficial for English script. However, the testing images were not refined by any preprocessing algorithm. So, optimizing images before entering the image to the detection procedure could have improving the localization results. Therefore, the preprocessing introduced by Nikolaou and Papamarkos (Nikolaou and Papamarkos 2009) reduces complex background colors, which is usually less than 15 colors. The practices in a 3D color histogram (usually no more than 100 colors) are clustered. Then a mean shift algorithm is applied to generate the final color layers to have solid characters and uniform local background. Therefore, a simple background can be implemented by a connected component technique or edge/gradient features to the segment text area in the image. Further, our text localization method using the calculation of bounding boxes

from the convex hull generation causes inaccurate ROIs in the text. The ROIs could be precisely located by using a powerful low-level detector called the Maximally Stable Extremal Region (MSER) proposed by (Matas et al. 2004), which has been widely explored in several recent projects (Nistér and Stewénius 2008, Neumann and Matas 2011, Neumann and Matas 2012, Shi et al. 2013, Koo and Kim 2013). The MSER algorithm is used as a method for blob detection in images that detects stable color regions, which provides a feasible location for localized text.

3) The improvement of text and non-text classification

The text localization process is intended to specify the area that is expected to be text, which receives coarse information based on the hypothesis that the candidate text region is a uniform pattern. In contrast, a non-text region is often formless because it contains many types of objects. Therefore, a classification process is important for text and non-text classification. We also presented classifying text and non-text in Chapter 3 and Chapter 4. The method in Chapter 3 exploits three object attributes (object description, color distribution, and gradient strength) to determine the objects characteristics, which provides useful results for classifying text and non-text. Further, a natural scene always includes a diversity of color; nevertheless, text is usually generated in a consistent and separable color which diverges from the background. By this supposition, classifying text and non-text by using color components could be helpful. So, we introduced the color autocorrelation histograms and combined these with the SIFT descriptors for text and non-text classification in Chapter 4. Our technique is robust for reflections, shadows and illumination variation because the autocorrelation functions (ACFs) for the histograms of color intensity can represent color information in an intensity-invariant manner. These methods could be adapted in object classification, video text localization as well as face recognition.

4) The improvement of character recognition

To recognize characters of a single font, we modified the SIFT matching function by matching keypoints of the testing images to the prototypical keypoints (PKPs) of the models based on both the descriptor and location as well as the model POIs. The recognition technique merely matches the keypoints of the testing images to

the PKP model without any intelligent algorithm, but the performances are still good. However, the larger the class, the lower the detection and recognition accuracy. One direction to improve the character recognition is using deep convolutional neural networks (deep CNN), a powerful tool for image representation by computing meaningful high-level deep features (Girshick et al. 2014, Krizhevsky et al. 2017), which could easily be used to classify a large number of character classes.

Bibliography

Alves, W. A. L. and Hashimoto, R. F.: 2010, Classification of regions extracted from scene images by morphological filters in text or non-text using decision tree, *18th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS (WSCG 2010)*, Václav Skala - UNION Agency, pp. 165–172.

Arthur, D. and Vassilvitskii, S.: 2007, K-means++: The advantages of careful seeding, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, Society for Industrial and Applied Mathematics, pp. 1027–1035.
URL: *http://dl.acm.org/citation.cfm?id=1283383.1283494*

Azad, P., Asfour, T. and Dillmann, R.: 2009, Combining harris interest points and the sift descriptor for fast scale-invariant object recognition, *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, IEEE Press, Piscataway, NJ, USA, pp. 4275–4280.
URL: *http://dl.acm.org/citation.cfm?id=1732643.1732747*

Beaudoin, N. and Beauchemin, S. S.: 2002, An accurate discrete fourier transform for image processing, *Object recognition supported by user interaction for service robots*, Vol. 3, pp. 935–939.

Bhattacharyya, A.: 1943, On a measure of divergence between two statistical populations defined by their probability distribution, *Bull. Calcutta Math. Soc* .
URL: *https://ci.nii.ac.jp/naid/10027606363/en/*

Borji, A., Cheng, M.-M., Jiang, H. and Li, J.: 2015, Salient object detection: A benchmark, *IEEE Transactions on Image Processing* **24**(12), 5706 – 5722.

Box, G. E. P. and Jenkins, G. M.: 1994, *Time Series Analysis: Forecasting and Control*, 3rd edn, Prentice Hall PTR, Upper Saddle River, NJ, USA.

Burt, P. J. and Adelson, E. H.: 1983, The laplacian pyramid as a compact image code, *IEEE Transactions on Communications* **31**(4), 532–540.

Cakebread, C.: 2017, People will take 1.2 trillion digital photos this year - thanks to smartphones.
URL: *https://www.businessinsider.com/12-trillion-photos-to-be-taken-in-2017-thanks-to-smartphones-chart-2017-8*

Canedo-Rodriguez, A., Kim, S., Kim, J. H. and Blanco-Fernandez, Y.: 2009, English to spanish translation of signboard images from mobile phone camera, *IEEE Southeastcon 2009*, pp. 356–361.

CCD (Image Sensor) Design: 2019.
URL: *http://av.jpn.support.panasonic.com/support/global/cs/dsc/knowhow/knowhow27.html*

Chen, X., Yang, J., Zhang, J. and Waibel, A.: 2004, Automatic detection and recognition of signs from natural scenes, *IEEE Transactions on Image Processing* **13**(1), 87–99.

Chiung-Yao, Chen, S.-W. and Fuh, C.-S.: 2003, Road-sign detection and tracking, *IEEE Transactions on Vehicular Technology* **52**(5), 1329–1341.

Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D. J. and Ng, A. Y.: 2011, Text detection and character recognition in scene images with unsupervised feature learning, *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, ICDAR '11, IEEE Computer Society, Washington, DC, USA, pp. 440–445.
URL: *http://dx.doi.org/10.1109/ICDAR.2011.95*

Collobert, R. and Bengio, S.: 2004, Links between perceptrons, mlps and svms, *ICML*.
URL: *https://doi.org/10.1145/1015330.1015415*

Cortes, C. and Vapnik, V.: 1995, Support-vector networks, *Machine Learning* **20**(3), 273–297.
URL: *http://dx.doi.org/10.1007/BF00994018*

Dalal, N. and Triggs, B.: 2005, Histograms of oriented gradients for human detection, *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, IEEE Computer Society, Washington, DC, USA, pp. 886–893.
URL: *http://dx.doi.org/10.1109/CVPR.2005.177*

de Campos, T. E., Babu, B. R. and Varma, M.: 2009, Character recognition in natural images, *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications*, Vol. 2, pp. 273–280.

Drews, P., de Bem, R. and de Melo, A.: 2011, Analyzing and exploring feature detectors in images, *9th IEEE International Conference on Industrial Informatics*, pp. 305–310.

Dumitras, T., Lee, M. L., Quinones, P., Smailagic, A., Siewiorek, D. P. and Narasimhan, P.: 2006, Eye of the beholder: Phone-based text-recognition for the visually-impaired, *2006 10th IEEE International Symposium on Wearable Computers*, pp. 145–146.

Epshtein, B., Ofek, E. and Wexler, Y.: 2010, Detecting text in natural scenes with stroke width transform, *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2963–2970.

Ezaki, N., Bulacu, M. and Schomaker, L.: 2004, Text detection from natural scene images: towards a system for visually impaired persons, *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 2, pp. 683–686 Vol.2.

Fan, L., Fan, L. and Tan, C. L.: 2001, Binarizing document image using coplanar prefilter, *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 34–38.

Farag, A. A.: 2014, *Biomedical Image Analysis: Statistical and Variational Methods*, Cambridge University Press.

Forgy, E. W.: 1965, Cluster analysis of multivariate data: Efficiency versus interpretability of classification, *Biometrics* **21**(3), 768–769.

Foulsham, T. and Kingstone, A.: 2017, Are fixations in static natural scenes a useful predictor of attention in the real world?, *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale* **71**, 172–181.

Fragoso, V., Gauglitz, S., Zamora, S., Kleban, J. and Turk, M.: 2011, Translatar: A mobile augmented reality translator, *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 497–502.

F.R.S., K. P.: 1900, X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **50**(302), 157–175.
URL: *https://doi.org/10.1080/14786440009463897*

Gevers, T. and Smeulders, A. W.: 1999, Color-based object recognition, *Pattern Recognition* **32**(3), 453 – 464.
URL: *http://www.sciencedirect.com/science/article/pii/S0031320398000363*

Girshick, R., Donahue, J., Darrell, T. and Malik, J.: 2014, Rich feature hierarchies for accurate object detection and semantic segmentation, *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, IEEE Computer Society, Washington, DC, USA, pp. 580–587.
URL: *https://doi.org/10.1109/CVPR.2014.81*

Goethe: 1982, *Theory of Colours, trans*, Cambridge, MA: MIT Press, Cambridge.

Gong, L., Feng, J. and Yang, R.: 2016, *Corner Detection-Based Image Feature Extraction and Description with Application to Target Tracking*, Springer India, New Delhi, pp. 1069–1076.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S. and Shet, V. D.: 2013, Multi-digit number recognition from street view imagery using deep convolutional neural

networks, *CoRR* **abs/1312.6082**.
URL: *http://arxiv.org/abs/1312.6082*

Hastie, T., Tibshirani, R. and Friedman, J.: 2009, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, second edition edn, Springer.

He, T., Huang, W., Qiao, Y. and Yao, J.: 2016, Text-attentional convolutional neural network for scene text detection, *Trans. Img. Proc.* **25**(6), 2529–2541.
URL: *http://dx.doi.org/10.1109/TIP.2016.2547588*

Herranz, L., Jiang, S. and Xu, R.: 2016, Modeling restaurant context for food recognition, *IEEE Transactions on Multimedia* **19**(2), 430–440.

Hsueh, M.: 2011, *Interactive text recognition and translation on a mobile device*, Master's thesis, EECS Department, University of California, Berkeley.
URL: *http://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-57.html*

Humphrey, G. K., Goodale, M. A., Jakobson, L. S. and Servos, P.: 1994, The role of surface information in object recognition: Studies of a visual form agnosic and normal subjects, *Perception* **23**(12), 1457–1481.

Hurvich, L. M. and Jameson, D.: 1957, An opponent-process theory of color vison, *Psychological Review* **64**(6), 384–404.

Ito, S. and Kubota, S.: 2010, Object classification using heterogeneous co-occurrence features, *Proceedings of the 11th European Conference on Computer Vision: Part II*, ECCV'10, Springer-Verlag, Berlin, Heidelberg, pp. 209–222.
URL: *http://dl.acm.org/citation.cfm?id=1888028.1888045*

Jaccard, P.: 1901, Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines., **37**, 241–72.

James, W.: 1890, *The Principles of Psychology*, Vol. 1, New York: Henry Holt.

Jian, M., Lam, K.-M., Dong, J., and Shen, L.: 2015, Visual-patch-attention-aware saliency detection, *IEEE Transactions on Cybernetics* **45**(8), 1575–1586.

Jiang, R., Qi, F., Xu, L. and Wu, G.: 2006, Detecting and segmenting text from natural scenes with 2-stage classification, *Sixth International Conference on Intelligent Systems Design and Applications*, Vol. 2, IEEE, pp. 819–824.

Joshi, A., Zhang, M., Kadmawala, R., Dantu, K., Poduri, S. and Sukhatme, G. S.: 2009, Ocrdroid: A framework to digitize text using mobile phones, *INTERNATIONAL CONFERENCE ON MOBILE COMPUTING, APPLICATIONS, AND SERVICES (MOBICASE)*, Springer-Verlag New York Inc, pp. 273–292.

Jung, J.-H., Lee, S.-H., Cho, M.-S. and Kim, J.-H.: 2011, Touch tt: Scene text extractor using touchscreen interface, *ETRI Journal* **33(1)**.

Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V. R., Lu, S., Shafait, F., Uchida, S. and Valveny, E.: 2015, Icdar 2015 competition on robust reading, *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1156–1160.

Khan, F. S., Anwer, R. M., Weijer, J., Bagdanov, A. D., Lopez, A. M. and Felsberg, M.: 2013, Coloring action recognition in still images, *Int. J. Comput. Vision* **105**(3), 205–221.
URL: *http://dx.doi.org/10.1007/s11263-013-0633-0*

Kim, K. C., Byun, H. R., Song, Y. J., Choi, Y. W., Chi, S. Y., Kim, K. K. and Chung, Y. K.: 2004, Scene text extraction in natural scene images using hierarchical feature combining and verification, *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 2, pp. 679–682 Vol.2.

Kim, K. I., Jung, K. and Kim, J. H.: 2003, Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**(12), 1631–1639.

Kingma, D. P. and Ba, J.: 2015, Adam: A method for stochastic optimization, *the 3rd International Conference for Learning Representations, San Diego, 2015*.

Kobayashi, T. and Otsu, N.: 2008, Image feature extraction using gradient local auto-correlations, *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, Springer-Verlag, Berlin, Heidelberg, pp. 346–358.

Koenderink, J. J.: 2010, *Color for the Sciences*, The MIT Press.

Koo, H. I. and Kim, D. H.: 2013, Scene text detection via connected component clustering and nontext filtering, *IEEE Transactions on Image Processing* **22**(6), 2296–2305.

Krizhevsky, A., Sutskever, I. and Hinton, G. E.: 2017, Imagenet classification with deep convolutional neural networks, *Commun. ACM* **60**(6), 84–90.
URL: *http://doi.acm.org/10.1145/3065386*

Kullback, S. and Leibler, R.: 1951, On information and sufficiency, *The Annals of Mathematical Statistics* **22**(1), 79–86.
URL: *http://mathfaculty.fullerton.edu/sbehseta/Kullback.pdf*

Kwok, N. M., Ha, Q. P. and Fang, G.: 2009, Effect of color space on color image segmentation, *2009 2nd International Congress on Image and Signal Processing*, pp. 1–5.

LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., uller, U. M., ackinger, E. S., Simard, P. Y. and Vapnik, V.: 1995, Comparison of learning algorithms for handwritten digit recognition, *International Conference on Artificial Neural Networks*, pp. 53–60.

Li, C., Ding, X. and Wu, Y.: 2001, Automatic text location in natural scene images, *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 1069–1073.

Li, Y., Jia, W., Shen, C. and van den Hengel, A.: 2014, Characterness: An indicator of text in the wild, *IEEE Transactions on Image Processing* **23**(4), 1666 – 1677.

Liang, J., Doermann, D. and Li, H.: 2005, Camera-based analysis of text and documents: A survey, *Int. J. Doc. Anal. Recognit.* **7**(2-3), 84–104.
URL: *http://dx.doi.org/10.1007/s10032-004-0138-z*

Lindeberg, T.: 1998, Feature detection with automatic scale selection, *International Journal of Computer Vision* **30**(2), 79–116.
URL: *http://dx.doi.org/10.1023/A:1008045108935*

Lloyd, S. P.: 2006, Least squares quantization in pcm, *IEEE Trans. Inf. Theor.* **28**(2), 129–137.
URL: *http://dx.doi.org/10.1109/TIT.1982.1056489*

Lowe, D. G.: 2004, Distinctive image features from scale-invariant keypoints, *Int. J. Comput. Vision* **60**(2), 91–110.
URL: *https://doi.org/10.1023/B:VISI.0000029664.99615.94*

Lucas, S. M.: 2005, Icdar 2005 text locating competition results, *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, Vol. 1, pp. 80–84.

Lucas, S. M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R., Ashida, K., Nagai, H., Okamoto, M., Yamamoto, H., Miyao, H., Zhu, J., Ou, W., Wolf, C., Jolion, J.-M., Todoran, L., Worring, M. and Lin, X.: 2005, Icdar 2003 robust reading competitions: Entries, results, and future directions, *Int. J. Doc. Anal. Recognit.* **7**(2-3), 105–122.
URL: *http://dx.doi.org/10.1007/s10032-004-0134-3*

Lucchese, L. and Mitra, S. K.: 2001, Colour segmentation based on separate anisotropic diffusion of chromatic and achromatic channels, *IEE Proceedings - Vision, Image and Signal Processing* **148**(3), 141–150.

Mapelli, D. and Behrmann, M.: 1997, The role of color in object recognition: Evidence from visual agnosia, *Neurocase* **3**(4), 237–247.
URL: *https://doi.org/10.1080/13554799708405007*

Matas, J., Chum, O., Urban, M. and Pajdla, T.: 2004, Robust wide-baseline stereo from maximally stable extremal regions, *Image and Vision Computing* **22**(10), 761 – 767. British Machine Vision Computing 2002.
URL: *http://www.sciencedirect.com/science/article/pii/S0262885604000435*

Matsukawa, T. and Kurita, T.: 2010, *Image Classification Using Probability Higher-Order Local Auto-Correlations*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 384–394.

McCulloch, W. S. and Pitts, W.: 1943, A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics* **5**(4), 115–133.

Mikolajczyk, K. and Schmi, C.: 2001, Indexing based on scale invariant interest points, *Eighth IEEE International Conference on Published in: Computer Vision (ICCV 2001)*, pp. 525–531.

Mikolajczyk, K. and Schmid, C.: 2004, Scale & affine invariant interest point detectors, *International Journal of Computer Vision* **60**(1), 63–86.
URL: *http://dx.doi.org/10.1023/B:VISI.0000027790.02288.f2*

Minetto, R., Thome, N., Cord, M., Leite, N. J. and Stolfi, J.: 2013, T-hog: An effective gradient-based descriptor for single line text regions, *Pattern Recognition* **46**(3), 1078 – 1090.
URL: *http://www.sciencedirect.com/science/article/pii/S0031320312004438*

Minetto, R., Thome, N., Cord, M., Leite, N. J. and Stolfi, J.: 2014, Snoopertext: A text detection system for automatic indexing of urban scenes, *Computer Vision and Image Understanding* **122**(Supplement C), 92 – 104.
URL: *http://www.sciencedirect.com/science/article/pii/S1077314213001914*

Neumann, L. and Matas, J.: 2011, Text localization in real-world images using efficiently pruned exhaustive search, *2011 International Conference on Document Analysis and Recognition*, pp. 687–691.

Neumann, L. and Matas, J.: 2012, Real-time scene text localization and recognition, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3538–3545.

Neumann, L. and Matas, J.: 2013, Scene text localization and recognition with oriented stroke detection, *2013 IEEE International Conference on Computer Vision*, pp. 97–104.

Nikolaou, N. and Papamarkos, N.: 2009, Color reduction for complex document images, *Int. J. Imaging Syst. Technol.* **19**(1), 14–26.

Nistér, D. and Stewénius, H.: 2008, Linear time maximally stable extremal regions, *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, Springer-Verlag, Berlin, Heidelberg, pp. 183–196.

Ohta, Y.-I., Kanade, T. and Sakai, T.: 1980, Color information for region segmentation, *ComputerGraphics and Image Processing* **13**(3), 222–241.

Otsu, N.: 1979, A threshold selection method from gray level histograms, *IEEE Transactions on Systems, Man and Cybernetics* **9**(1), 62–66.

Pan, Y.-F., Hou, X. and Liu, C.-L.: 2009, Text localization in natural scene images based on conditional random field, *2009 10th International Conference on Document Analysis and Recognition*, pp. 6–10.

Pan, Y.-F., Hou, X. and Liu, C.-L.: 2011, A hybrid approach to detect and localize texts in natural scene images, *IEEE Transactions on Image Processing* **20**(3), 800–813.

Panhwar, M. A., Memon, K. A., Abro, A., Zhongliang, D., Khuhro, S. A. and Memon, S.: 2019, Signboard detection and text recognition using artificial neural networks, *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 16–19.

Park, J., Yoon, H. and Lee, G.: 2007, Automatic segmentation of natural scene images based on chromatic and achromatic components, *in* A. Gagalowicz and W. Philips (eds), *Computer Vision/Computer Graphics Collaboration Techniques*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 482–493.

Patel, C. I., Patel, A. and Patel, D.: 2012, Optical character recognition by open source ocr tool tesseract: A case study, *International Journal of Computer Applications* **55**(10), 50–56.

Petter, M., Fragoso, V., Turk, M. and Baur, C.: 2011, Automatic text detection for mobile augmented reality translation, *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 48–55.

Redmon, J. and Farhadi, A.: 2018, Yolov3: An incremental improvement, *CoRR* **abs/1804.02767**.
URL: *http://arxiv.org/abs/1804.02767*

Rosenblatt, F.: 1958, The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review* **65**(6), 386–408.

Saeidi, M., Saeidi, K. and Khaleghi, M.: 2008, Noise reduction in image sequences using an effective fuzzy algorithm, *International Journal of Computer and Information Engineering* **2**(7), 2540–2545.

Salton, G. and McGill, M. J.: 1986, *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., New York, NY, USA.

Shapiro, L. and Stockman, G.: 2001, *Computer Vision*, Prentice Hall.

Sharma, G., Jurie, F. and Schmid, C.: 2012, Discriminative spatial saliency for image classification, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3506–3513.

Shi, B., Bai, X. and Belongie, S. J.: 2017, Detecting oriented text in natural images by linking segments, *CoRR* **abs/1703.06520**.
URL: *http://arxiv.org/abs/1703.06520*

Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S. and Zhang, Z.: 2013, Scene text recognition using part-based tree-structured character detection, *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2961–2968.

Sikka, A. and Wu, B.: 2012, Camera based equation solver for android devices.
URL: *https://www.scribd.com/document/238796927/Sikka-Wu-Automatic-Equation-Solver-and-Plotter*

Simonyan, K. and Zisserman, A.: 2014, Very deep convolutional networks for large-scale image recognition, *3rd International Conference on Learning Representations*.

Smith, R., Gu, C., Lee, D.-S., Hu, H., Unnikrishnan, R., Ibarz, J., Arnoud, S. and Lin, S.: 2016, End-to-end interpretation of the french street name signs dataset, *Computer Vision - ECCV 2016 Workshops* **9913**, 411–426.

Smolka, B., Plataniotis, K., Chydzinski, A., Szczepanski, M., Venetsanopoulos, A. and Wojciechowski, K. W.: 2002, Self-adaptive algorithm of impulsive noise reduction in color images, *Pattern Recognition* **35**(8), 1771 – 1784. Colour Imaging.
URL: *http://www.sciencedirect.com/science/article/pii/S0031320301001698*

Song, X., Jiang, S., Xu, R. and Herranz, L.: 2014, Semantic features for food image recognition with geo-constraints, *2014 IEEE International Conference on Data Mining Workshop*, pp. 1020–1025.

Sriman, B. and Schomaker, L.: 2015a, Explicit foreground and background modeling in the classification of text blocks in scene images, *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 755–759.

Sriman, B. and Schomaker, L.: 2015b, Object attention patches for text detection and recognition in scene images using sift, *ICPRAM 2015 - 4th International Conference on Pattern Recognition Applications and Methods, Proceedings*, Vol. 1, pp. 304–311.

Su, B., Tian, S., Lu, S., Dinh, T. A. and Tan, C. L.: 2013, Self learning classification for degraded document images by sparse representation, *12th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 155–159.

Tanaka, J. W. and Presnell, L. M.: 1999, Color diagnosticity in object recognition, *Perception & Psychophysics* **61**(6), 1140–1153.

Torp, H., Kristoffersen, K. and Angelsen, B. A. J.: 1994, Autocorrelation techniques in color flow imaging: signal model and statistical properties of the autocorrelation estimates, *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* **41**(5), 604–612.

Trémeau, A., Fernando, B., Karaoglu, S. and Muselet, D.: 2011, Detecting text in natural scenes based on a reduction of photometric effects: Problem of text detection, *in* R. Schettini, S. Tominaga and A. Trémeau (eds), *Computational Color Imaging*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 230–244.

Troost, J. M. and Weert, C. M. M. D.: 1991, Naming versus matching in color constancy, *Perception & Psychophysics* **50**(6), 591–602.
URL: *http://dx.doi.org/10.3758/BF03207545*

Tsotsos, J. K., Culhane, S. M., Wai, W. Y. K., Lai, Y., Davis, N. and Nuflo, F.: 1995, Modeling visual attention via selective tuning, *Artificial Intelligence* **78**(1-2), 507–545.
URL: *http://dx.doi.org/10.1016/0004-3702(95)00025-9*

van Boven, B., van der Putten, P., Åström, A., Khalafi, H. and Plaat, A.: 2018, Real-time excavation detection at construction sites using deep learning, *Advances in Intelligent Data Analysis XVII*, pp. 340–352.

Vidakovic, V. and Zdravković, S.: 2009, Color influences identification of the moving objects more than shape, *Psihologija* **42**(1), 79–93.

Walther, D., Rutishauser, U., Koch, C. and Perona, P.: 2004, On the usefulness of attention for object recognition, *Workshop on Attention and Performance in Computational Vision at ECCV*, pp. 96–103.

Wang, K. and Belongie, S.: 2010, Word spotting in the wild, *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, Springer-Verlag, Berlin, Heidelberg, pp. 591–604.
URL: *http://dl.acm.org/citation.cfm?id=1886063.1886108*

Wang, W., Luo, D. and Li, W.: 2009, Algorithm for automatic image registration on harris-laplace features, *Journal of Applied Remote Sensing* **3**(1), 033554–033554–13.
URL: *http://dx.doi.org/10.1117/1.3256135*

Weixing, W., Ting, C., Sheng, L. and Enmei, T.: 2015, Remote sensing image automatic registration on multi-scale harris-laplacian, *Journal of the Indian Society of Remote Sensing* **43**(3), 501–511.
URL: *http://dx.doi.org/10.1007/s12524-014-0432-2*

Wu, J.-C., Hsieh, J.-W. and Chen, Y.-S.: 2008, Morphology-based text line extraction, *Machine Vision and Applications* **19**(3), 195–207.
URL: *http://dx.doi.org/10.1007/s00138-007-0092-0*

Wu, J., Cui, Z., Sheng, V. S., Zhao, P., Su, D. and Gong, S.: 2013, A comparative study of sift and its variants, *Measurement Science Review* **13**(3), 122–131.

xcorr Cross-correlation: 2017.
URL: *https://www.mathworks.com/help/signal/ref/xcorr.html#bucjo5f*

Xiang, X., Lv, N., Guo, X., Wang, S. and Saddik, A. E.: 2018, Engineering vehicles detection based on modified faster r-cnn for power grid surveillance, *Sensors* **18**(7), 2258.

Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., and Zhang, Z.: 2015, The application of two-level attention models in deep convolutional neural network for fine-grained image classification, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 842–850.

Xu, J., Shivakumara, P., Lu, T., Phan, T. Q. and Tan, C. L.: 2014, Graphics and scene text classification in video, *2014 22nd International Conference on Pattern Recognition*, pp. 4714–4719.

Yao, C., Bai, X., Liu, W., Ma, Y. and Tu, Z.: 2012, Detecting texts of arbitrary orientations in natural images, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1083–1090.

Yi, C. and Tian, Y.: 2011, Text detection in natural scene images by stroke gabor words, *2011 International Conference on Document Analysis and Recognition*, pp. 177–181.

Zhang, J., Chen, Q., Bai, X., Sun, Q., Sun, H., Zhang, D. X. J., Chen, Q., Bai, X., Sun, Q., Sun, H. and Xia, D.: 2009, An advanced harris-laplace feature detector with high repeatability, *Image and Signal Processing, 2009. CISP '09. 2nd International Congress on*, pp. 1–5.

Zhang, W. and Su, T.: 2016, Reference beam pattern design for frequency invariant beamforming based on fast fourier transform, *Sensors (Basel, Switzerland)* **16**(10).

Zhou, L., Zhou, Z. and Hu, D.: 2013, Scene classification using a multi-resolution bag-of-features model, *Pattern Recognition* **46**(1), 424 – 433.
URL: *http://www.sciencedirect.com/science/article/pii/S0031320312003330*

Zhu, A., Wang, G. and Dong, Y.: 2015, Detecting natural scenes text via auto image partition, two-stage grouping and two-layer classification, *Pattern Recognition Letters* **67**(Part 2), 153 – 162. Granular Mining and Knowledge Discovery.
URL: *http://www.sciencedirect.com/science/article/pii/S0167865515001713*

Zhu, S. and Zanibbi, R.: 2016, A text detection system for natural scenes with convolutional feature learning and cascaded classification, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 625–632.

Summary

This thesis deals with the challenge of text recognition in scene images, i.e., photographs, of the man-made environment. The automatic recognition of such texts would provide a wide range of useful applications, ranging from general location-dependent information support, scene-text translation for tourists, up to support systems of visually disabled persons. However, it appears that in order to solve this goal, many image problems need to be addressed for which no satisfactory solution exists, thus far. Therefore we explore algorithms for detection and classification of text in the environment, by applying several image-feature approaches. A basic tenet is that the essential features enabling selective attention to scene text are both gradient and color based.

Chapter 1 provides a brief general introduction to visual attention for text localization and classification follow by geometrical problems of image processing and background of image processing on mobile devices. Further in Chapter 1, the research questions, a survey of recent publications in the field, and the objectives of the dissertation are explained.

Chapter 2 introduces SIFT-based modeling of character objects for scene-text detection and recognition. The proposed technique could encourage a robot to investigate and recognize the characters in a natural scene, which usually contains a variety of objects and potential obstacles. In this way, the top-down scheme, which is an expectancy-driven character model with attentional patches, responds to the requirements that the process should be reasonably fast and able to handle variable sizes and fonts. The type of character modeling serves two purposes: detec-

tion and recognition. The character models are constructed from cropped grayscale and binary (B/W) character images, from which features are extracted by the 128-dimensions of SIFT descriptors. The model of each character feature is built yielding a codebook of prototypical keypoints (PKP) by k-means.

The models (object attention patches) are evaluated regarding their individual provisory character recognition performance. Character recognition results for different values of codebook size show that a higher number of PKPs improves the accuracy, although at the cost of memory. When applying SIFT with the attentional patch approach to classify a subset of similar-shaped characters, the incorrect classification is decreased. The results show that our proposed model-based approach can be applied for a coherent SIFT-based text detection and recognition process both in English and non-English scripts.

Chapter 3 proposes a classification technique for classifying foreground (FG) and background (BG) based on the bag of visual words (BOVW) involving the object attribute types. For the textual information in a real-world scene, it is hard to predict multiple text patterns, orientation or colors against a background because of the cluttered background. Individual FG and BG zones are detected for points of interest (POI) by the Harris-Laplace (HL) detection method because it provides a more significant portion of salient image patches than SIFT. Next, the histogram of three object attribute types (object description (SIFT), color distribution (CDIS), and gradient strength (GLAC)) are extracted. A codebook approach is used for each attribute type using clustering collaboration with the SVM in order to generate the classifier. To get an optimal distance for the codebooks, the experimental results show that the city block algorithm is suited to generate SIFT and GLAC models. Then a cosine algorithm is appropriated for the CDIS model.

The classification rate for text by SiftSVM is higher than the other two classifiers (CDisSVM and GLACSVM classifiers). Meanwhile, the consideration of the color distribution using the CDisSVM classifier gives medium accuracy. So, color distribution (CDis) could be improved by combining it with another feature. The new classifier OppSiftSVM, which is formed by SIFT and CDIS, increases the classification rate. Based on the advantage of the combined features, the two classifiers, OppSiftSVM and GLACSVM, are employed to classify the observation feature as text or non-text with the voting weights. Classifying text and non-text by OppSiftSVM and

GLACSVM improves the accuracy rate. The experiments show that our method performs effectively in foreground and background classification in comparison with existing algorithms.

Chapter 4 presents a novel robust feature-based classification of scene image blocks based on the optimal hybrid combination of two features: the color autocorrelation histogram (CAH) and the scale invariant feature transform (SIFT) algorithm. SIFT is an efficient local feature descriptor yielding scale, orientation, and light variation. Due to the complex relation between lighting conditions and reflective properties of text (FG) and background (BG), autocorrelation describes the correlation between data sequences in a single series of sample values, at several delays in 'time'. Therefore, a color-based feature is contrived to represent color information in an intensity-invariant manner, by using the autocorrelation functions (ACFs) for the histograms of color intensity. To obtain the codebooks or bag of visual word (BOVW), all the feature vectors are grouped into clusters using the k-means algorithm.

The optimum control parameters such as patch sizes, distances, etc. to study the impact of different conditions for SIFT and CAH are defined. The accuracy of FG/BG classification with a 1NN classifier using the various types of color spaces tested on three datasets shows that the color schemes RGB and YUV perform well on the Chars74K and TSIB datasets, while the performance on the ICDAR2015 benchmark data set is slightly lower. The regular nearest neighbor (1NN) and the support-vector machine (SVM) are compared. The experiments show that 1NN gave a better result against the SVM for both CAH and SIFT features, but the 1NN performs relatively slower than the SVM when there are large codebooks, due to the matching of all the keypoints to the codebook PKPs. However, the results for the SVM depend on the model parameters, in particular, C determining the level of misclassified results. The performances of the proposed model appear to be robust and suitable for Asian scripts such as Kannada and Thai, where high curvature and salient color variations characterize the scene text fonts.

Chapter 5 promotes a new continuous word script localization in a scene image using a top-down scheme based on the object expectancy approach, with a faster computation and similar to human behavior. Many proposed text localization methods are usually based on Western languages, in which the elements of words differ

from Asian scripts such as Thai. For example, Thai sentences do not have spaces between words. Therefore, a model of continuous-word script detection in natural scene images is more interesting and challenging. In the case of missing detection, a classification method using the model of general text-chunk and non-text classifiers based on the SIFT feature is very important. The codebook-based classifiers are constructed according to the bag of visual words (BOVW) concept.

In the stage of text localization based on the expectancy approach, we expect that the text codebook should match all the texts in the testing image while the non-text model should match the complex background of the image. After the matching process, all the matched keypoints of text are plotted on a 2D spatial image. To identify the detected area, the adjacent keypoints are merged by the dissolving method. The horizontal line is merged by motion blur, while the Gaussian blur is applied to the vertical line and the dissolved image is complete. The dissolved image is then taken by a convex hull algorithm to have bounding boxes of the actual text areas or the candidate regions of interest (ROI). Finally, the candidate ROIs are classified as text or non-text by the object classifiers. The results show that our proposed technique improves the text localization, particularly for a continuous-word script (Thai) which is rich in ornaments. The proposed method is also beneficial for locating the text of the ICDAR2015 dataset even though there are many text lines in an image.

Future research on the basis of this dissertation can be directed at an improved diversity of script styles. We have shown that a specific script such as Thai requires an adapted processing variant to be effective. The same can be expected of other international scripts. The localization of text blocks needs further attention because its success determines the usability of subsequent optical-character recognition stages. Finally, the type of detected regions presents challenges to the classification layer, which warrants special attention in embedded applications, such as smart-phone apps.

Samenvatting

Deze dissertatie gaat over het detecteren en herkennen van teksten op foto's in 'scene-afbeeldingen', meestal foto's die genomen zijn met een smart phone, binnen de bebouwde omgeving. De automatische herkenning van dergelijke teksten heeft een breed spectrum van nuttige toepassingen, varierend van locatie-afhankelijke informatiediensten, straatnaam en -wegwijzer vertaling voor toeristen die een taal of schrift niet machtig zijn, tot ondersteuningssystemen voor visueel gehandicapten. Ook de herkenning van naambordjes binnen een gebouw kan nuttig zijn bij de ondersteuning van navigatie gedurende het laatste deel van een traject ('de laatste paar honderd meter'). Het blijkt echter dat er nog veel problemen zijn waarvoor tot nu toe geen bevredigende oplossing bestaat. Daarom onderzoeken we algoritmen voor de detectie en herkenning van tekstbeelden in de omgeving, door gebruik te maken van verschillende methodes uit de automatische beeldbewerking en machineleren. Een basisprincipe dat wordt gehanteerd is dat tekstbeelden, zelfs in verschillend schrifttypen, vorm- en kleureigenschappen in hebben die afwijken van natuurlijke beeldonderdelen. Hierdoor kan een systeem selectieve aandacht richten op tekstonderdelen om deze vervolgens te kunnen herkennen.

Hoofdstuk 1 geeft een korte algemene inleiding op het probleem visuele aandacht voor tekstlokalisatie en -classificatie. Een overzicht wordt gegeven van geometrische problemen binnen de beeldverwerking en achtergronden van beeldverwerking op mobiele apparaten. Bovendien worden in hoofdstuk 1, de onderzoeksvragen gepresenteerd. Een overzicht van recente publicaties in het veld wordt gegeven en de doelstellingen van het proefschrift worden uitgelegd.

Hoofdstuk 2 introduceert een aangepaste, SIFT-gebaseerde modellering van karakter-objecten voor de opsporing van de tekstbeelden en hun automatische herkenning. De voorgestelde techniek is breder toepasbaar dan alleen tekstherkenning, maar zou ook door autonome robots kunnen worden gebruikt. Het algoritme gebruikt namelijk 'top-down' verwachtingen over de statistisch meest waarschijnlijke layout van bekende individuele beeldonderdelen, 'attentional patches'. Een autonoom systeem moet snel kunnen omgaan met verschillende afmetingen en lettertypen. In het algoritme wordt een individueel karakter beschouwd als een meetkundige vorm met een centrum en daaromheen een aantal informatieve beeldelementen. Deze beeldelementen ('patches') zijn opgebouwd uit bijgesneden grijswaarden en binaire (z/w) afbeeldingen, waaruit kenmerken worden gextraheerd, m.n. de 128 numerieke kenmerken van de SIFT descriptor. Het model van de van elk teken omvat (a) een visueel code-boek van prototypische aandachtspunten (PKP), die berekend worden door k-means clustering en (b) een genormaliseerd statistisch model met de verwachte posities van deze elementen. Met deze methode kan de detectie en de herkenning worden gecombineerd. De experimenten onderzoeken verschillende afmetingen van een dergelijk code-boek en tonen aan dat een groter aantal PKPs de nauwkeurigheid verbetert. Dit kost echter wel meer geheugen. Wanneer men SIFT toepast samen met de voorgesteld aanpak van 'attentional patches' kunnen de herkenningsresultaten worden verbeterd. De resultaten tonen ook aan dat het voorgestelde model kan worden toegepast voor zowel westerse als niet-westerse schriftsoorten zoals Thais schrift.

Hoofdstuk 3 presenteert een detectietechniek voor de indeling in voorgrond (FG) en achtergrond (BG) op basis van de tas van de visuele woorden (BOVW). In plaats van alleen en eenvoudig op helderheid en kleur af te gaan is het uitgangspunt dat de statistische signaalbronnen voor voorgrond (tekst) en de achtergrond (natuurlijke of urbane contextbeelden) afzonderlijk moeten worden gemodelleerd. Voor de tekstuele informatie in een levensechte scene is het met eenvoudige methoden moeilijk te voorspellen waar de tekst is gelokaliseerd vanwege de vaak rommelige achtergrond. Individuele FG en BG zones worden gedetecteerd door 'points of interest' (POI) te detecteren. Hierbij bleek dat de detectiemethode Harris-Laplace (HL) betere resultaten oplevert dan SIFT, omdat er veel meer punten van belang worden gedetecteerd in het beeld. Vervolgens is echter een expliciete modellering

nodig van het beeld rond elk aandachtspunt. Hiervoor werd een combinatie van methoden gebruikt, die zowel vormelementen ('edges') als kleur omvatten. Drie objectkenmerktypen werden gebruikt: beschrijving van het object (waarvoor SIFT dan weer wel bruikbaar is), kleurverdeling (CDIS) en kleurovergangen (GLAC)). Een code-boek wordt voor een optimale combinatie, leidend tot een histogram dat als kenmerksvector gebruikt kan worden voor classificatie met een SVM. Bij het bepalen van een optimale afstandsmaat voor de code-boeken, blijkt uit de experimentele resultaten dat de Manhattan afstand ('city block') het meest geschikt is voor de SIFT en GLAC modellen. Bij de toepassing van het CDIS-model, werd de cosinus-similariteit van vectoren gebruikt. Voor de individuele methoden geldt dat SiftSVM beter presteert dan de andere twee classificaties (CDisSVM en GLACSVM). Wanneer kleur wordt meegenomen verbetert de prestatie (CDisSVM). Onze nieuwe classificatiemethode echter, OppSiftSVM, die wordt gevormd door de SIFT en CDIS, verhoogt de voorgrond/achtergrond classificatie echter het duidelijkst.

Hoofdstuk 4 presenteert een nieuwe robuuste detectie van tekstblokken in een foto. Het probleem is dat kleurinformatie, hoewel belangrijk, onderhevig is aan allerlei variatie, zoals belichting. Een nieuwe methode voor kleurinvariantie wordt voorgesteld, die gebruik maakt van autocorrelatie van het kleurhistogram (CAH). Deze representatie kon weer gecombineerd worden met vorminformatie uit de SIFT-representatie en kaner een 'bag of visual words' (BOVW) berekend worden. De nauwkeurigheid van voorgrond/achtergrond classificatie is getest met verschillend kleurruimtes, op drie datasets. De resultaten tonen dat de kleurenschema's RGB- en YUV goed op twee bekende datasets presteren (Chars74K en TSIB), terwijl de prestaties op de ICDAR2015 benchmark een iets lagere prestatie toonden. De reguliere 'nearest neighbor' (1NN) en de support-vector machine (SVM) werden vergeleken. De experimenten tonen aan dat de 1NN een beter resultaat gaf dan de SVM, voor zowel CAH en SIFT kenmerken, maar de 1NN is relatief trager. De resultaten voor de SVM hangen sterk af van de parameters van het model, in het bijzonder de parameter C. De prestaties van het voorgestelde model blijken robuust te zijn en geschikt voor Aziatische scripts zoals Kannada en Thai, waar het lettertype vaak wordt gekenmerkt door hoge kromming en opvallende kleurvariaties (zeker bij reclameteksten in het straatbeeld).

Hoofdstuk 5 bevordert een nieuwe continue woord script lokalisatie in een beeld

van de scne met behulp van een top-down-regeling gebaseerd op de object levensverwachting aanpak, met een snellere berekening en vergelijkbaar met menselijk gedrag. Veel voorgestelde tekst lokalisatie methoden zijn meestal gebaseerd op westerse talen, waarin de elementen van woorden van Aziatische scripts zoals Thais verschillen. Thaise zinnen hebben bijvoorbeeld geen spaties tussen woorden. Daarom is een model voor continu-word script detection in natuurlijke scne beelden meer interessant en uitdagend. In het geval van ontbrekende detectie is een indelingsmethode met behulp van het model van algemene tekst-stuk en niet-tekstuele classificaties op basis van de SIFT-functie erg belangrijk. De code boek gebaseerde classificaties zijn gebouwd volgens de tas van de visuele woorden (BOVW) concept. In het stadium van de lokalisatie van de tekst gebaseerd op de aanpak van de levensverwachting, verwachten wij dat de tekst code boek moet overeenkomen met alle teksten in het testen beeld terwijl de niet-tekst-model moet overeenkomen met de complexe achtergrond van de afbeelding. Na de matching-proces, worden alle overeenkomende aandachtspunten van tekst uitgezet op een ruimtelijke 2D-afbeelding. Voor de identificatie van het gedetecteerde gebied zijn worden de aangrenzende aandachtspunten samengevoegd door de oplossen methode. De horizontale lijn is samengevoegd door bewegingsonscherpte, terwijl de Gaussiaanse vervaging wordt toegepast op de verticale lijn en de opgeloste image voltooid is. De opgeloste afbeelding vervolgens onderneemt een convex hull algoritme dat de omsluitende kaders van de werkelijke tekstgebieden of de kandidaat-regiǒ2019s van belang (ROI). Ten slotte, de kandidaat-ROIs worden geclassificeerd als tekst of niettekst door de classificaties van het object. De resultaten tonen aan dat onze voorgestelde techniek verbetert de lokalisatie van de tekst, met name voor een continue-word script (Thai), die rijk aan ornamenten is. De voorgestelde methode is ook gunstig voor het lokaliseren van de tekst van de dataset ICDAR2015, zelfs al zijn er veel tekstregels in een afbeelding.

Toekomstig onderzoek op basis van dit onderzoek kan worden gericht op het kunnen verwerken van een nog hogere diversiteit aan schriftstijlen. We hebben aangetoond dat een specifiek schrift, zoals Thai, een aangepaste verwerking vereist om effectief te kunnen worden gedetecteerd en herkend. Hetzelfde kan worden verwacht van andere (complexe) internationale schrifttypen. De lokalisatie van tekstblokken behoeft verdere aandacht, omdat het succes hiervan de bruikbaarheid van

latere optische tekenherkenning stadia bepaalt. De herkenning van tekstbeelden met een aantal gangbare technieken zelf, gaf tot slot verhoudingsgewijs gunstige resultaten. Meer onderzoek is nodig, maar met de ontworpen methoden kunnen nu in ieder geval grote 'training sets' worden verzameld om 'OCR' systemen te trainen. Tot slot valt te vermelden dat de keuzen die gemaakt zijn in dit onderzoek, in hoge mate bepaald zijn door de specifieke beperkingen die gesteld worden aan embedded toepassingen ('apps') in huidige smart phones.

Publications

Publications by the author

Journal Paper

1. Bowornrat Sriman, Lambert Schomaker (2019) "*Multi-script text versus non-text classification of regions in scene images*", Journal of Visual Communication and Image Representation (JVCI), Elsevier Science, Vol. 62, pp. 23-42.

Conference Papers

1. Bowornrat Sriman, Lambert Schomaker (2015) "*Explicit foreground and background modeling in the classification of text blocks in scene images*", Proc. of 3rd Int. Conf. on IAPR Asian Conference on Pattern Recognition (ACPR), pp. 755-759, 3-6 November, 2015, Kuala Lumpur, Malaysia.

2. Bowornrat Sriman, Lambert Schomaker (2015) "*Object attention patches for text detection and recognition in scene images using SIFT*", Proc. of 4th Int. Conf. on Pattern Recognition Applications and Methods, Vol. 1, pp. 304-311, 10-12 January, 2015, Lisbon, Portugal.

3. Bowornrat Sriman, Lambert Schomaker, Potchara Pruksasri (2016) "*General Text-Chunk Localization in Scene Images using a Codebook-based Classifier*", Proc. of 4th Int. Conf. on IIAE Intelligent Systems and Image Processing (ICISIP2016), pp. 134-141, 8-12 September, 2016, Kyoto, Japan.

Acknowledgments

The Ph.D. study is the first overseas trip in my life and for the second time to live on my own after an undergraduate internship. Seeing something new, different cultures, including public transports that support all kinds of people, makes me so excited and impressible. So I would like to thank those who gave me a chance to get many perspectives on many different things throughout my Ph.D. journey.

First of all, I really appreciate my supervisor, Prof.Dr.Lambert Schomaker who provided me with the opportunity to pursue my study at the University of Groningen. You are quite understanding about the characteristic of Asian people and also gave me the time when I needed advice. Many times you have suggested me to learn new techniques to get the effective results of the research. Sometimes I may not be able to do, but you are always patient and encouraging me. You are not only introduced much useful software but also explained how to use a material that related to my research. I would like to big thank my co-supervisor Asst.Prof.Dr.Chatklaw Jareanpon who answered every question I ever asked her. I would also like to thank Asst.Prof.Dr.Marco Wiering for giving me some helpful advice.

Another people that are indispensable to be especially thankful is Nuffic who gave me the scholarship during my education through Wiebe Zijlstra (the coordinator international cooperation of NFP) and the project assistant International Cooperation (Gonny Lakerveld and Esme Bakker). I would like to acknowledge Asst.Prof.Dr.Kasom Chanawong, the president of College of Asian Scholars, who supported me to study abroad. Thanks to Qu Lei, who adjusted the letters to request funding. I would also thank you to Claire Taylor, who proofread the research

papers. I would also like to thank my committee members (Prof.Dr.Gernot A. Fink, Prof.Dr.D. Gavrila, and Prof.Dr.Raffaella Carloni) for their help with my thesis and their unique perspectives.

I would like to thank faculty secretaries Elina Sietsema, Jessica Rutgers, Carlijne de Vries, Jan van Hoogen, and Sarah van Wouwe for all your assistance. I want to thank Harm Vos, Remco Wouts, and the Center for Information Technology, the University of Groningen for their support on this research. Additionally, I would like to thank all staff members in Artificial Intelligence Department who have opened their mind and friendly.

My thanks to the first two colleagues in the office Jean-Paul van Oosten for helping me to translate many documents in Dutch and Amir Shantia for setting a network printer on my laptop and telling me the forecast website that can alert the rain every two hours because there is rainy almost every day in the Netherlands. Thank you to Sheng He for sharing the office and playing "Go" with me. Thanks to Michiel Holtkamp, Gyuhee and Ahmed Maruf for sharing the office and discussing different cultures between European and Asian countries. Many thanks to Harmen de Weerd, who taught me some statistic topic. Special thanks to my sister Pornntiwa Pawara, who let me stay with you for four days, before I am going back to Thailand. You are always very kind/talkative/cheerful, which made me feel relax and also thank you for helping me to fix some coding.

Special thanks to my paranimfen Pornntiwa Pawara and Ahmed Maruf who helped in the PhD ceremonial preparation. I want to thank my friends in Artificial Intelligence Department Charlotte, Jean-Paul van Oosten, Amir Shantia, Faik Karaaba, Olarik Surinta, Trudy Buwalda, Harmen de Weerd, Ioanna Katidioti, Burcu Arslan, Marijke Beulen, Jort Bergfeld, Sheng He, Mahya Ameryan, Pornntiwa Pawara, Ahmed Maruf, Ana Bosnic, Emmanuel Okafor, Ben Wolf, and Stefan Huijser for all the lunches, sharing their experiences, and friendship with me. My apologies if I miss someone's name.

Asst.Prof.Dr.Phatthanaphong Chomphuwiset, thank you for your help to recommend and adjust in some contents of my research paper. I would like to grateful the Informatics Mahasarakham University who provided the computers to run the experiments and workplace when I did my research in Thailand. Thanks to my colleagues Yuwarat Thitinirankul, Sangsom, and Wachira Chankhaikhot for help-

ing me to coordinate the college during my study. My dataset could not be accomplished without persons who help me to annotate the Thai characters for TSIB dataset, so, thank you to Rattiyapon Rungthong, Patchara, Nunnipha Prasunluk, Alongkot Pruksasri, and Alongkorn Singhalha.

My time in the Netherlands would have been much more difficult without Thai friends, Siriluk Rakphong, Oranath Van Der Weide, Jane Kajuiter, Dollaya Nantalohit, Uma Pupphachai, Duenpen Unjaroen, Naim Laeni, Theerut Luangmonkong, and many others. I am grateful to you all that we had several activities together. Many thanks to Satiya Phongsakun and Dennis Stuijfzand who help me to translate the summary section of my thesis from English to Dutch.

Lastly, I am especially thankful to my parents, Samruan Sriman and Sunanthorn Sriman and my husband's parents, Atsadong Pruksasri and Sujitra Pruksasri for the incredible amount of support and encouragement. I am also thankful to my husband, Potchara Pruksasri, who always cheered me up and supported me to make my research successful.

Bowornrat Sriman
Groningen
February 10, 2020